НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут телекомунікаційних систем

Кафедра Телекомунікаційних систем

«На правах рукопису»
УДК 621.39

«До захисту допущено»
Завідувач кафедри
_____ Л.О. Уривський
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 172 Телекомунікації та радіотехніка

на тему: «Дослідження можливостей IMS для надання нових послуг в інфокомунікаційних мережах при реалізації концепції FMC»

Виконав:
студент ІІ курсу, групи ТС-81м
Смолій Дмитро Анатолійович _____

Керівник:
доцент, кандидат технічних наук
Гаттуров В.К. _____

Рецензент:
доцент, кандидат технічних наук
Мазор С.Ю. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем

Кафедра Телекомунікаційних систем

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (спеціалізація) – 172 «Телекомунікації та радіотехніка» (172.3620.1 «Телекомунікаційні системи та мережі»)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Л.О. Уривський

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Смолію Дмитру Анатолійовичу

1. Тема дисертації «Дослідження можливостей IMS для надання нових послуг в інфокомунікаційних мережах при реалізації концепції FMC», науковий керівник дисертації Гаттуров Віктор Кавіч, доцент, кандидат технічних наук, затверджені наказом по університету від «___»_____ 20__ р. №_____
2. Термін подання студентом дисертації 9 грудня 2019 року.
3. Об’єкт дослідження мережі NGN на базі мультимедійної IP-підсистеми.
4. Предмет дослідження можливості підсистеми IMS для надання нових послуг у хмарному середовищі.
5. Перелік завдань, які потрібно розробити
   - дослідження інноваційних змін в інфокомунікаційних мережах при їх еволюційному переході до концепції мультимедійної IP підсистеми;
   - огляд сучасних хмарних технологій, моделей їхнього обслуговування та рішень віртуалізації мереж IMS;
   - огляд адаптивної моделі багатоетапної подійно-орієнтованої архітектури SEDA та базових рішень щодо створення програмного забезпечення на її основі;
   - створення логічної моделі домашнього серверу абонентів та дослідження процесу надходження і обробки великої кількості запитів про надання послуг, що надходять від абонентів мережі IMS до нього.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу
Плакат №1 «Тема, мета, актуальність, об’єкт, предмет, проблематика, завдання дослідження»
Плакат №2 «Вертикальні сервісні платформи, горизонтальні сервісні платформи»
Плакат №3 «Архітектура мережі IMS, архітектура мережі оператора зв’язку вз провадженою підсистемою IMS»
Плакат №4 «Перехід від традиційних мереж до NFV, схема експериментальної моделі процесу прийому та обробки заявок»
Плакат №5 «Система моніторингу PabbitMQ під час проведення експерименту з реалізацією архітектури SEDA. Графічний інтерфейс виконуваної програми»
Плакат №6 «Висновки»

7. Перелік публікацій
Гаттуров В. К., Смолій Д.А. ДОСЛІДЖЕННЯ МОЖЛИВОСТЕЙ IMS ДЛЯ НАДАННЯ НОВИХ ПОСЛУГ В ІНФОКОМУНІКАЦІЙНИХ МЕРЕЖАХ ПРИ РЕАЛІЗАЦІЇ КОНЦЕПЦІЇ FMC. ПРІТС 2019. - К.: КПІ ім. Ігоря Сікорського, 2019.

8. Дата видачі завдання 1 вересня 2018 року.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Огляд літературних джерел по тематиці роботи	01.09.18 - 01.11.18	
2.	Формування мети та наукових завдань дослідження	01.11.18 - 01.12.18	
3.	Формування структури роботи та наповнення теоретичної частини	01.01.19 - 15.03.19	
4.	Підготовка матеріалів для доповіді на конференції	15.03.19 - 16.04.19	
5.	Формування вступної частини роботи	18.04.19 - 15.05.19	
6.	Проведення досліджень для вирішення наукових завдань	15.05.19 - 01.09.19	
7.	Формування останнього розділу роботи, висновків	01.09.19 - 01.11.19	
8.	Оформлення плакатів та пояснювальної записки дипломної роботи	01.11.19 - 01.12.19	
9.	Підготовка до захисту дипломної роботи	01.12.19 - 16.12.19	

Студент Смолій Д.А.

Науковий керівник дисертації Гаттуров В.К.

РЕФЕРАТ

Темою магістерської дисертації є дослідження можливостей IMS для надання нових послуг в інфокомунікаційних мережах при реалізації концепції FMC

Робота містить 124 сторінки, зокрема 50 рисунків, 14 таблиць та 20 джерел інформації.

Актуальність теми обумовлена розвитком та застосуванням мереж наступного покоління (NGN) та їх конвергенцією з мобільними мережами 4G LTE і як наслідок, утворення єдиної мережі з реалізацією мультимедійної IP підсистеми, окремі функціональні об’єкти якої розташовуються у хмарному середовищі.

Мета даної роботи полягає у підтвердженні можливостей надання нових послуг абонентам інфокомунікаційної мережі, при реалізації архітектури IMS у хмарному середовищі.

Об’єкт дослідження реалізація компонентів IMS у хмарному середовищі.

Предмет дослідження продуктивність роботи окремих компонентів мультимедійної IP підсистеми, що реалізовані у хмарному середовищі на базі SEDA, при їх обробці вхідного потоку заявок.

В даній роботі з’ясовується роль мультимедійної IP підсистеми та її зв’язок з мобільними мережами 4G LTE. Розглядаються моделі обслуговування концепції хмарних технологій та способи реалізації віртуалізації мережевих функцій. Визначається шаблон створення програмного забезпечення на основі багатоетапної подійно-орієнтованої архітектури SEDA. Проводиться експериментальне дослідження для з’ясування продуктивності роботи домашнього серверу абонентів, реалізованого у хмарному середовищі.

КОНЦЕПЦІЯ, АРХІТЕКТУРА, IP MULTIMEDIA SUBSYSTEM, 4G LTE, ХМАРНІ ТЕХНОЛОГІЇ, NETWORK FUNCTION VIRTUALIZATION, ПОСЛУГА, STAGED-EVENT DRIVEN ARCHITECTURE

ABSTRACT

The relevance of the topic is due to the development and deployment of Next Generation Networks (NGNs) and their convergence with 4G LTE mobile networks and, as a consequence, the formation of a single network with the implementation of multimedia IP subsystem, separate functional objects of which are located in a cloud environment.

The purpose of this work is to confirm the possibility of providing new services to subscribers of the information and communication network, while implementing the IMS architecture in a cloud environment.

The object of the study is the implementation of IMS components in a cloud environment.

The subject of the study is the performance of the individual components of the multimedia IP subsystem implemented in the SEDA-based cloud environment when processing the incoming requests stream.

This paper explores the role of the IP multimedia subsystem and its connection to 4G LTE mobile networks. Service models of concepts cloud technologies and ways of virtualization realization of network functions are considered. Defines a pattern for creating software based on staged-event driven architecture. An experimental study is being conducted to determine the performance of a home subscriber server implemented in a cloud environment.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

AMQP	Advanced Message Queuing Protocol
AMR	Adaptive Multi Rate
AS	Application Server
BGCF	Breakout Gateway Control Function
CS	Chanel switching
EMS	Element Management System
ETSI	European Telecommunications Standards Institute
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GW	Gateway
I-CSCF	Interrogating Call Session Control Function
IM-SSF	IP Multimedia Service Switching Function
IP	Internet Protocol
IT	Information Technology
laaS	Infrastructure as a Service
MAC	Medium Access Control
MANO	Management and Orchestration
MEGACO	Media Gateway Control Protocol
MME	Mobility Management Entity
MMTel	IMS Multimedia Telephony
MRF	Media Resource Function
MRFC	Media Resource Function Controller
MRFP	Media Resource Function Processor
NFVI	NFV Infrastructure
NFVO	NFV Orchestrator
OSA	Open Service Access
OSS	Operation Support System

OTT	Over-The-Top Service
PaaS	Platform as a Service
P-CSCF	Proxy CSCF
PDF	Policy Decision Function
PS	Packet Switching
PSTN	Public Switched Telephone Network
SaaS	Software as a Service
SCS	Service Capability Server
S-CSCF	Serving Call Session Control Function
SLF	Subscriber Location Function
SSF	Service Switching Function
TAS	Telephony Application Server
TISPAN	Telecommunications and Internet converged Services and Protocols for Advanced Networking
VIM	Vi Improved
vIMS	Virtual IMS
VoLTE	Voice over LTE
VoWiFi	Voice over Wireless Fiber
ОС	Операційна система
ПЗ	Програмне забезпечення
СУБД	Система управління базами даних

ВСТУП

У наш час оператори зв'язку надають велику кількість різних послуг, голосовий зв'язок все ще приносить для них більшу частину доходів. Але в той же час між провайдерами йде активна боротьба за користувачів, так як ринок послуг мобільного зв'язку поступово переходить в стан насичення, темпи зростання абонентської бази операторів значно знижуються, тобто подальше залучення абонентів можливе тільки за рахунок переходу абонентів від одного оператора до іншого, але не за рахунок підключення нових. Кількість клієнтів і отримання від них прибутку безпосередньо залежить від якості послуг, що надаються, а це означає, що провайдерам необхідно шукати нові шляхи для вирішення даного завдання.

Тому на ринку відбуваються активні процеси злиття і поглинання. Нерідко консолідуються компанії - провайдери різних типів послуг (фіксована і мобільна телефонія, мобільна телефонія і кабельне телебачення і т.п.), в той же час різке зростання споживання мобільного трафіку даних, гостра конкуренція і високий попит на послуги мобільного широкосмугового доступу вимагає впровадження нових і дорогих технологій. Розвиток мереж 4G, а також їх взаємозв’язок зі стаціонарними мережами, в свою чергу, стимулює операторів до впровадження концепції IMS (IP Multimedia Subsystem).

Концепція IMS була розроблена в 2002 р. консорціумом 3rd Generation Partnership Project (3GPP) для мереж 3G/WCDMA(Wideband Code Division Multiple Access) і стандартизована в специфікаціях 3GPP R.5. Пізніше створеною моделлю скористалися відразу декілька провідних інфокомунікаційних організацій, які визначили IMS як основну мережну інфраструктуру наступного покоління. IMS забезпечує архітектуру в якій багато функцій можуть бути використані з різними додатками і у різних провайдерів, що, в свою чергу, дозволяє швидко і ефективно створювати нові послуги і безпосередньо надавати їх. В основі концепції цього стандарту лежить здатність IMS передавати сигнальний трафік, використовуючи

сигнальний протокол SIP (Session Initiation Protocol) і трафік в каналі через IP рівень, а також виконувати функції маршрутизатора або механізм керування сесіями абонентів з використанням інформації про їх стан.

Таким чином технологія IMS допомагає провайдерам об'єднувати всі типи мереж в одну, реалізувати комплекс послуг і сервісів, що поєднує в собі можливості мобільного і фіксованого зв'язку на базі однієї платформи (конвергентні послуги) і забезпечити операторам збільшення доходів, оскільки вона дає можливість впроваджувати нові типи послуг, в тому числі і в сегменті передачі голосу на мережах 4G LTE: VoLTE (Voice over Long-Term Evolution), VoWiFi (Voice over Wireless Fidelity).

Однак, навіть при переході операторів до роботи своїх мереж на основі концепції IMS не зменшує їхніх витрат на обслуговування, ремонт, заміну та купівлю нового обладнання, адже кількість трафіку, який обертається у світі, зростає з кожним днем.

Універсальним виходом з даної ситуації стала хмарна технологія віртуалізації мережевих функцій NFV (Network Functions Virtualization). У цих умовах NFV дозволяє оператору позбавлятися від спеціалізованого, пропрієтарного і дорогого в обслуговуванні обладнання, замінюючи його віртуальними пристроями, тобто повністю програмними рішеннями на універсальних серверах. Ті чи інші мережеві функції реалізуються віртуально (програмуються) на універсальному, масовому, отже, недорогому обладнанні, наприклад на стандартних серверах виробництва HP, Juniper, Dell або IBM, завдяки чому оператори можуть значно знизити різноманітність використовуваного обладнання, а отже, оптимізувати витрати на його придбання і експлуатацію.

Так як наразі перенесення функціонування елементів мережі до хмарного середовища не стандартизовано, інфокомунікаційні оператори можуть користуватися будь-якими програмними засобами для реалізації своїх потреб. Одним з таких засобів є багатоетапна подійно-орієнтована архітектура SEDA (Staged-event driven architecture).

Дана концепція дозволяє створювати програмне забезпечення, в якому керування доступними ресурсами здійснюється шляхом управління певними компонентами, які носять назву етап або стадія. Так уже готовий продукт буде мати вигляд системи пов’язаних між собою етапів або ж стадій, що робить його зручним для реалізації у хмарному середовищі.

У даній роботі буде проведено експериментальне дослідження із використанням концепції SEDA. Так буде створено модель домашнього серверу абонентів на який надходить велика кількість вхідних заявок про надання послуг від користувачів оператора послуг, що дозволить провести перевірку продуктивності роботи даного віртуалізованого елементу у мережі.

РОЗДІЛ 1. СТРУКТУРА ТА РОЛЬ КОНЦЕПЦІЇ IMS В СУЧАСНИХ ІНОФОКОМУНІКАЦІЙНИХ МЕРЕЖАХ

1.1 Виникнення концепції IMS

В сучасному світі, який постійно розвивається людство все більше відходить від аналогових технологій до цифрових. Це викликано тим, що старі аналогові системи не можуть забезпечити всі вимоги абонентів по об’єму, швидкості і якості передачі даних. Але нюансом також є те, що деякі абоненти не хотіли переходити на нові способи передачі даних. Рішенням даної проблеми стали мережі Next Generation Network (NGN).

Мережа NGN – універсальна мережа, яка підтримує як абонентів, які використовують термінали нового покоління, так і абонентів традиційних мереж зв’язку. В основу концепції NGN закладена ідея про створення універсальної мережі, яка б дозволяла переносити будь-які види інформації, такі як: мова, відео, аудіо, графіку і т.п., а також забезпечити можливість представлення необмеженого спектру інфокомунікаційних послуг.

Однак, розвиток мобільних мереж зв’язку і здешевлення, та відповідно, поширення серед суспільства мобільних телефонів викликало стрімкий ріст обсягів мобільного трафіку, що включає в себе як і передачу голосової інформації, так і передачу даних. Але головним мінусом даних мереж було і є те, що передача будь-якого виду інформації в них відбувається за допомогою технології з комутацією каналів. Тобто для кожного абонента мережі при створенні певного сеансу зв’язку створюється окремий канал для надання йому викликаної послуги, що в свою чергу не може забезпечити хорошу якість передачі голосової інформації, або ж велику швидкість передачі даних, на відміну від пакетних мереж передачі даних, на основі яких будується концепція мережі NGN. Звичайно, за отримання мобільних послуг і за користування послугами мережі NGN користувач платить різним операторам, які між собою, зазвичай, ніяк не пов’язані, окремо. Це викликає незручності як і для

звичайного користувача інфокомунікаційних послуг, так і для гігантів інфокомунікаційного ринку.

З точки зору звичайної людини, вона хоче мати доступ до всіх необхідних їй сервісів в будь-якому місці, з будь-якого з доступних для неї пристроїв. А оператори послуг, хочуть надавати своїм абонентам якнайширший спектр послуг і відповідно, не ділити можливу грошову вигоду між собою.

Так виникло бажання про інтегрування можливостей мереж NGN у мережі мобільного зв’язку, що в свою чергу викликало появу концепції технології IP Multimedia Subsystem, яка, грубо кажучи, дозволяє здійснити перехід від концепції надання послуг TriplePlay (голос, відео, дані) до 4Play (голос, відео, дані + мобільність) шляхом об’єднання стаціонарних та мобільних мереж в одну єдину систему. Таким чином забезпечується втілення в життя концепції Fiber Mobile Convergence (FMC).

IMS, в свою чергу, дозволяє розробляти і надавати абонентам мереж фіксованого та мобільного зв'язку персоналізовані послуги, засновані на різних комбінаціях голосових послуг, тексту, графіки та відео (чат на екрані мобільного телефону, електронна пошта, ігри та багато чого іншого).

Як наслідок, рішення IMS значно розширюють можливості кінцевого користувача за рахунок надання розширеного набору послуг, в тому числі тих, які були неможливі або економічно неефективні в мережах NGN на Softswitch. [1]

1.1.1 Забезпечення конвергенції послуг та роль сигнального протоколу SIP в мережах NGN

Розглянемо реалізацію конвергенції послуг на прикладі сучасної технології стаціонарних мереж - NGN.

Мережа NGN – універсальна мережа, яка підтримує як абонентів, які використовують термінали нового покоління, так і абонентів традиційних мереж зв’язку. В основу концепції NGN закладена ідея про створення

універсальної мережі, яка б дозволяла переносити будь-які види інформації, такі як: мова, відео, аудіо, графіку і т.п., а також забезпечити можливість представлення необмеженого спектру інфокомунікаційних послуг.

Що б подолати проблеми взаємодії мереж з різними системами сигналізації застосовується гнучкий комутатор Softswitch.

Даний пристрій, який іменується також як гнучкий комутатор, забезпечує взаємодію протоколів технологічно різних мереж на рівні сигнальних повідомлень. Так, для надійного переносу сигнальної інформації, гнучкий комутатор взаємодіє з рядом шлюзів MG, які знаходяться поблизу від джерел і приймачів інформації в ТфЗК (телефонія загального користування) (на границях мережі IP). Взаємодія зазвичай забезпечується при наявності, принаймні двох сигнальних шлюзів SG (Signalin Gateway), в які включені сигнальні ланки ОКС7(Система сигналізації №7).

Протокол SIP, який є складовою ОКС7, зазвичай працює поверх протоколу IP, розмова користувачів розглядається як мультимедійний сеанс зв’язку, який включає в себе передачу.

Softswitch, або ж MGC (Media Gateway Controller), використовується як сполучна ланка між ТфЗК і мережами SIP, так як виклики із телефонної мережі можуть надходити на IP–телефони і навпаки, а виклики із ТфЗК, в свою чергу, можуть проходити транзитом через мережу SIP.

Для взаємодії протоколу SIP з телефонними мережами загального користування було розроблено опис протоколу SIP, який отримав назву SIP-T.

Специфікації SIP-T (SIP for Telephony) описують способи передачі загальноканальної сигналізації ОКС-7 по мережі SIP, для чого використовується два алгоритми переносу сигналізації, відомі як інкапсуляція і трансляція.

Так, взаємодія мереж з комутацією пакетів і комутацією каналів за допомогою Softswitch засновано на інкапсуляції повідомлень мережі з комутацією каналів в тіло запитів SIP і трансляції частини інформації цих повідомлень , необхідної для правильної маршрутизації, в заголовок запиту SIP.

Для повідомлень мережі з комутацією каналів, які проходять через мережу SIP, трансляція дозволяє таким елементам мережі SIP, як проксі–сервери, котрі не вміють працювати безпосередньо з повідомленнями мереж з комутацією каналів, правильно передавати повідомлення, засновуючись на даних, трансльованих із повідомлення мережі з комутацією каналів в заголовок запиту SIP (наприклад , номер абонента, який викликається).

1.1.2 Поява концепції IMS, як крок до розвитку мереж NGN

Створення архітектури NGN відіграло важливу роль у новітній історії світових телекомунікацій. Але сучасний світ не стоїть на місці, що викликало необхідність створити новий підхід до організації різноманітних телекомунікаційних послуг для будь-якого користувача і в будь-якому місці мережі NGN. Цей новий підхід, званий IMS, виник в результаті еволюції мереж, побудованих на базі технології Softswitch, до якої була додана область управління мультимедійними сеансами на базі протоколу SIP та широкий спектр мультимедійних послуг, включаючи, зрозуміло, двосторонній аудіо/відеозв'язок.

Втім, принципово нова концепція IMS в значній мірі виявилася новою віхою у житті ідеї інтелектуальної мережі, перенесеної в світ конвергентних мобільних/фіксованих комунікацій. Більш того, спочатку дана концепція закладалась як просте перенесення архітектури Softswitch на мобільні мережі зі збереженням все тих же трьох рівнів:

- User Plane - призначений для користувача рівень, або рівень передачі даних;
- Control Plane - рівень управління;
- Application Plane - рівень додатків.

Як і в обох прообразах (Інтелектуальна мережа і Softswitch), в концепціях IMS специфікуються не вузли мережі, а функції, які зв'язуються один з одним через стандартизовані еталонні точки, подібно способу концепції кінця

минулого століття - концепції ISDN (Integrated Services Digital Network). Як і в архітектурі ISDN, еталонні точки в IMS використовуються для ідентифікації та опису інтерфейсів між різними функціональними мережевими елементами.

Гнучкий комутатор Softswitch і концепція IMS мають схожі архітектурні компоненти: рівень доступу, транспорт, управління викликами/сесіями, сервери додатків. Сигнальним протоколом, що «об'єднує» дані дві архітектури є протокол SIP, що дозволяє втілити ідею надання всіх послуг на базі IP-мережі. Проте, є деякі розбіжності.

Ідеологія Softswitch зародилась в епоху фіксованих (а згодом - конвергентних) телекомунікаційних мереж, що працюють в середовищах Time Division Multiplexing (TDM) і IP, тому значна увага в ній зосередилося на управлінні шлюзами по протоколу Megaco/H.248.

З іншого боку, для архітектури IMS, яка повністю орієнтується на IP, авторство якої належить мобільному партнерству 3GPP, проблема медіашлюзів зовсім не так актуальна.

До складу IMS входять три основні блоки:

- Функція управління сеансами зв'язку CSCF (Call Session Control Function) – яка виконує обов'язки SIP-сервера;
- База даних користувачів HSS (Home Subscriber Server), яка є аналогом HLR в мережах GSM;
- Медіасервер MS (Media Server), який є аналогом інтелектуальної периферії в концепції Інтелектуальної мережі і являє собою мережеве обладнання, яке може спільно використовувати кілька різних програм і забезпечує надання таких загальних мультимедійних послуг, як перетворення тексту в мову TTS (Text-To-Speech), відеосервіси, автоматичне розпізнавання мови, відеоконференц-зв'язок і так далі. [2]

Розглянемо детальніше архітектуру мереж IMS.

1.2 Архітектура IMS мереж

Для IMS розроблена багаторівнева архітектура з поділом транспорту для перенесення трафіку і сигнальної мережі IMS для управління сеансами, яку можна побачити на рисунку 1.2.1. 3GPP при розробці IMS фактично переніс на мобільні мережі основну ідеологію Softswitch. Хоча деякі функції не завжди легко віднести до того чи іншого рівня, але такий підхід забезпечує мінімальну залежність між рівнями.

У IMS можна виділити три рівні її архітектури:

• User Plane - призначений для користувача рівень або рівень передачі даних;

• Control Plane - рівень управління;

• Application Plane - рівень додатків.

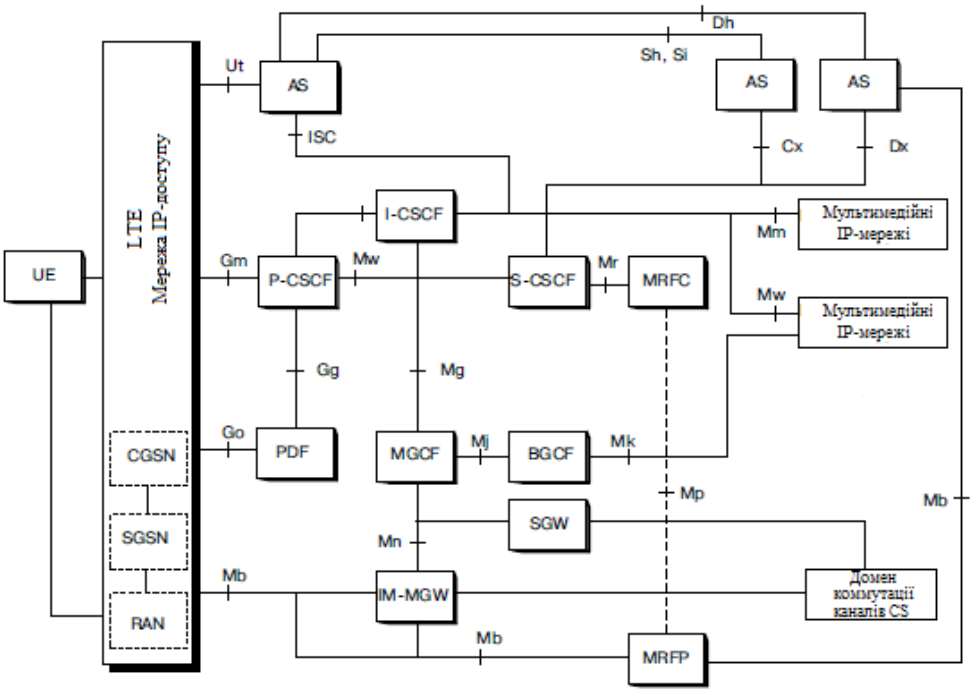

Рисунок 1.2.1 Архітектура мережі IMS

Важливою особливістю IMS є те, що організація 3GPP специфікує не вузли мережі, а функції. Це означає, що IMS-архітектура, як і Інтелектуальна мережа або гнучкий комутатор Softswitch, теж являє собою набір функцій,

з'єднаних стандартними інтерфейсами. Розробники мають право комбінувати кілька функцій в одному фізичному об'єкті або, навпаки, реалізувати одну функцію розподіленою, однак найчастіше фізичну архітектуру ставлять у відповідність функціональної і реалізують кожну функцію в окремому вузлі.

Різні функції IMS зв'язуються один з одним через набір контрольних точок, подібно способу, який був розроблений ще в концепції ISDN. Як і в архітектурі ISDN, еталонні точки в IMS використовуються для ідентифікації та опису інтерфейсів між різними функціональними мережевими елементами. Функції Call Session Control Function (CSCF), які знаходяться в центрі рисунку 1.2.1 поміщені туди не випадково.

1.2.1 Функціональні можливості IMS

У IMS застосовано новий підхід до надання послуг, що дозволяє оператору впроваджувати послуги, створені сторонніми розробниками або навіть самим оператором, а не виробниками телекомунікаційного обладнання. Це дозволяє інтегрувати різні послуги і надає широкі можливості персоналізації і збільшення кількості послуг.

В даний час, до впровадження IMS, в мобільних мережах використовуються так звані «вертикальні сервісні платформи», які успішно справляються з наданням невеликого числа ключових послуг.

Підхід IMS передбачає горизонтальну архітектуру (Рисунок 1.2.1.1), що дозволяє оператору просто і економічно впроваджувати нові персоналізовані послуги, причому користувачі можуть в рамках одного сеансу зв'язку отримати доступ до різних послуг.

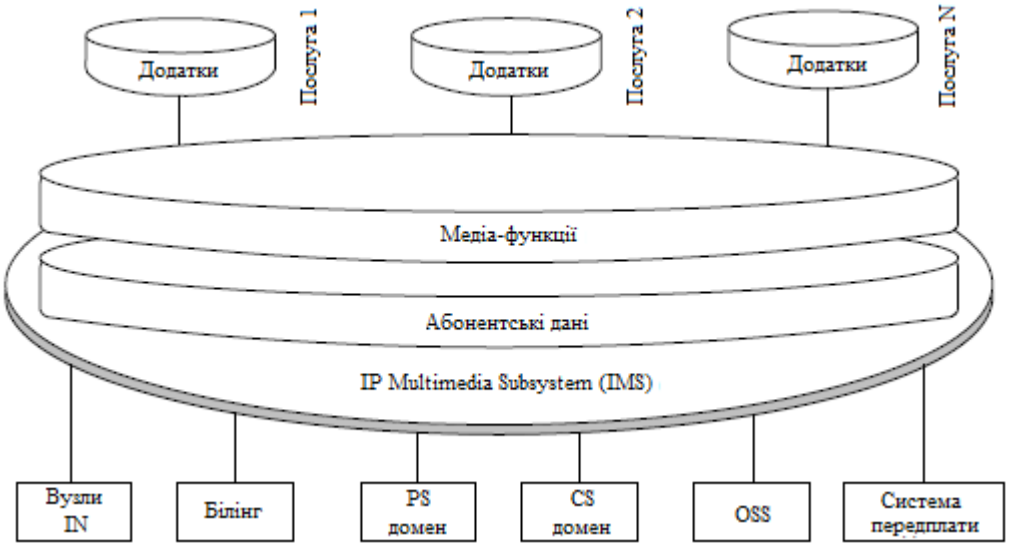

Рисунок 1.2.1.1 Горизонтальні сервісні платформи

1) Мультимедійні IP-сеанси

IMS може надавати широкий спектр послуг, але одна з них безумовно зберігає провідну роль - двосторонній аудіо / відео зв'язок. Для цього архітектура IMS повинна підтримувати сеанси мультимедійного зв'язку в IP мережах, причому такий зв'язок має бути доступний користувачам як в домашній, так і в гостьовій мережі. Мультимедійний зв'язок був стандартизований вже в ранніх документах 3GPP, ще до появи IMS, але надавався тільки доменом комутації каналів.

2) Якість обслуговування

Підтримка QoS (Quality of Service) є фундаментальним вимогою до IMS. При організації сеансу користувача устаткування сповіщає IMS про свої можливості і свої вимоги до QoS. За допомогою протоколу SIP можливо врахувати такі параметри, як тип і напрямок передачі даних, бітова швидкість, розмір пакетів, використання RTP (Real-time Transport Protocol), необхідна ширина смуги пропускання. IMS дозволяє управляти якістю зв'язку, яку отримає той чи інший користувач, і таким чином диференціювати користувачів і послуги. [3]

3) Взаємодія з іншими мережами

Функція підтримки взаємодії з мережею Інтернет очевидна, тому що завдяки спільним протоколам користувачі IMS можуть встановлювати мультимедійні сеанси зв'язку з різними службами глобальної мережі. Оскільки перехід від NGN та IMS буде поступовим і більш-менш тривалим, то IMS повинна також мати можливість взаємодії з мережами попередніх поколінь - стаціонарними (ТфЗК) і мобільними (2G/3G) мережами з комутацією каналів. Функції взаємодії з мережами комутації каналів не мають, зрозуміло, далекої перспективи, але вони абсолютно необхідні протягом довгого періоду існування конвергентних мереж.

4) Інваріантність доступу

Функціональні можливості IMS інваріантні щодо різних технологій доступу до неї, відмінних від GPRS (General Packet Radio Service), наприклад, використовують WLAN (Wireless Local Area Network), xDSL (Digital Subscriber Line), HFC (Hybrid Fiber Coax), кабельний модем і т.п. Як і будь-яка IP-мережа, IMS інваріантна щодо протоколів нижніх рівнів і технологій доступу. Функції доступу в IMS відокремлені від ядра мережі, тому інваріантність доступу до IMS отримала назву IP connectivity access і передбачає застосування будь-якої технології доступу, яка може забезпечити транспортування IP-трафіку між призначеним для користувача обладнанням і об'єктами IMS.

5) Створення послуг та управління послугами

Необхідність швидко впроваджувати різноманітні послуги, оскільки саме вони повинні стати основним джерелом доходів оператора в 21 столітті, зажадала переглянути процес створення послуг в IMS. Щоб зменшити час впровадження послуги і забезпечити її надання в гостьовій мережі, коли користувач знаходиться в роумінгу, в IMS ведеться стандартизація не послуги, а можливостей надання послуг (service capability). Таким чином, Оператор може впровадити будь-яку послугу, відповідну service capability, причому ця послуга буде підтримуватися і при переміщенні користувача в гостьову мережу, якщо ця мережа має аналогічними стандартизованими service capability. [3]

6) Роумінг

Функції роумінгу існували вже в мобільних мережах 2G, 3G, і IMS, природно, ці функції успадкувала. Однак саме поняття «роумінг» тепер істотно розширилося і включає в себе:

• GPRS-роумінг - гостьова мережа надає RAN (Radio access network) і SGSN (Serving GPRS Support Node), а в домашній знаходяться GGSN і IMS;

• IMS-роумінг - гостьова мережа надає IP-з'єднання і точку входу (наприклад P-CSCF (Proxy CSCF)), а домашня мережа забезпечує всі інші функції;

• CS-роумінг (Chanel switching) - роумінг між мережею IMS і мережею комутації каналів.

7) Безпека

Функції забезпечення безпеки необхідні кожній телекомунікаційній системі, і IMS надає рівень безпеки, не менший, ніж GPRS-мережі та мережі комутації каналів. IMS проводить аутентифікацію користувачів перед початком надання послуги, надає користувачеві можливість запросити конфіденційність інформації, що передається під час сеансу, та інше.

8) Нарахування плати

IMS дозволяє Оператору або провайдеру послуг гнучко призначати тарифи для мультимедійних сеансів. IMS зберігає можливість нараховувати плату за сеанс найбільш простим способом - в залежності від тривалості сеансу або від обсягу трафіку, але може також використовувати більш складні схеми, що враховують різну призначену для користувача політику, компоненти медіа-даних, послуги що надаються тощо. Потрібно також, щоб дві IMS-мережі при необхідності могли обмінюватися інформацією, потрібною для нарахування плати за сеанс зв'язку. IMS підтримує нарахування плати в режимі як online, так і offline. [3]

1.2.2 Функціональні вузли IMS

1) Call Session Control Function (CSCF).

CSCF – по суті головний управляючий компонент архітектури IMS - підрозділяється на три області відповідальності: взаємодія, обслуговування і управління проксі сеансом. Відповідно до концепції IMS, кожна сигнальна подія, яку генерує користувач, спершу направляється до проксі-CSCF (P-CSCF) незалежно від самої сигнального події, яка може передбачати такі речі як запит на встановлення сеансу зв'язку, активізацію необхідної функції, виділення мережевого ресурсу, запит обслуговування іншим додатком. Таким чином, функція P-CSCF є першим з ким зустрічається користувач в IMS-ядрі.

Отримавши повідомлення SIP від пристрою користувача, P-CSCF пересилає їх функції до I-CSCF (Interrogating Call Session Control Function) або функції S-CSCF (Serving Call Session Control Function). Функція I-CSCF служить єдиною точкою реєстрації в мережі для доступу до послуг IMS як для місцевих, так і роумінгових користувачів. Як тільки I-CSCF реєструє користувача, S-CSCF вступає до роботи і починає процес управління сеансом зв'язку, забезпечуючи доступ до всіх необхідних служб.

Важливо відзначити, що в більшості випадків I-CSCF поводиться як SIP-проксі сервер незважаючи на те, що її основна роль в IMS полягає у визначенні місця надання послуг.

Підкреслимо, що може бути багато I-CSCF в межах мережі постачальника послуг, які відповідальні за наступні функції:

- реєстрація, яка є процесом призначення S-CSCF;
- управління потоком сеансу, що використовується для того, щоб направити SIP запит, який було отримано з іншої мережі, до S-CSCF, або направити запити SIP від користувачів на різні S-CSCF;
- створення даних в формі Call Detail Records (CDR) для білінгу, що генерують дохід від використання ресурсу.

Насправді, S-CSCF виконує роль проксі-сервера SIP, який сеанс за сеансом виконує управління входженням в зв'язок абонентів мережевої служби. Він керує взаємодією з базами даних мережі, такими як домашній абонентський сервер HSS для підтримки мобільних користувачів і серверів аутентифікації, авторизації та обліку AAA (Authentication, Authorization and Accounting). У цій ролі S-CSCF також передає сигнальний трафік, що генерується користувачами роумінгу (тобто, користувачами, які під'єднані до мережі в якості гостей) в їхні домашні мережі, де зберігається профіль абонентів і звідки надсилаються підтвердження їх платоспроможності (Billability). Як тільки S-CSCF виконав свої обов'язки перевірки користувача, запит може бути переданий відповідним серверам додатків і шлюзів, які потрібні для організації запитуваного сеансу. Якщо запит призначається для іншої мережі, то цілком ймовірно, що доведеться застосувати інші протоколи сигналізації, здійснити перетворення протоколу, організувати взаємодію з іншими типами медіа в інших мережах.

Центральним для роботи IMS є P-CSCF, причому в міру розвитку архітектури IMS його обов'язки трансформувались. Спочатку P-CSCF ніс відповідальність за функцію вибору політики обслуговування PDF (Policy Decision Function), яка накопичує, зберігає, управляє і звертається до політик, щоб прийняти рішення, пов'язані із запитами розподілу ресурсів IP. У міру розвитку архітектури IMS, після низки дискусій PDF була виведена зі сфери P-CSCF, щоб зробити її більш доступною для бездротових LAN і інших мереж доступу. Що ж стосується елементів P-CSCF, то в силу їх ролі «воротаря» і того факту, що вони ідеально підходять для збору даних про сесіях, P-CSCF генерують записи білінгу, які можуть бути накопичені і передані централізованій функції Charging Gateway Function (CGF) перед генерацією користувальницьких рахунків. [4]

2) Бази користувачів HSS і SLF

Кожна IMS-мережа містить один або більше серверів користувацьких баз даних HSS. По суті, HSS є централізованим сховищем інформації про абонентів і послуги та в свою чергу є еволюційним розвитком HLR з архітектури мереж

GSM (Global System for Mobile Communications). У HSS зберігається вся інформація, яка може знадобитися при встановленні мультимедійного сеансу: інформація про місцезнаходження користувача, інформація для забезпечення безпеки (аутентифікація і авторизація), інформація про профіль користувача, про обслуговуючу користувача S-CSCF, і про тригерні точки звернення до послуг. Функції, що виконуються HSS, показані в загальному вигляді на рисунку 1.2.2.1.

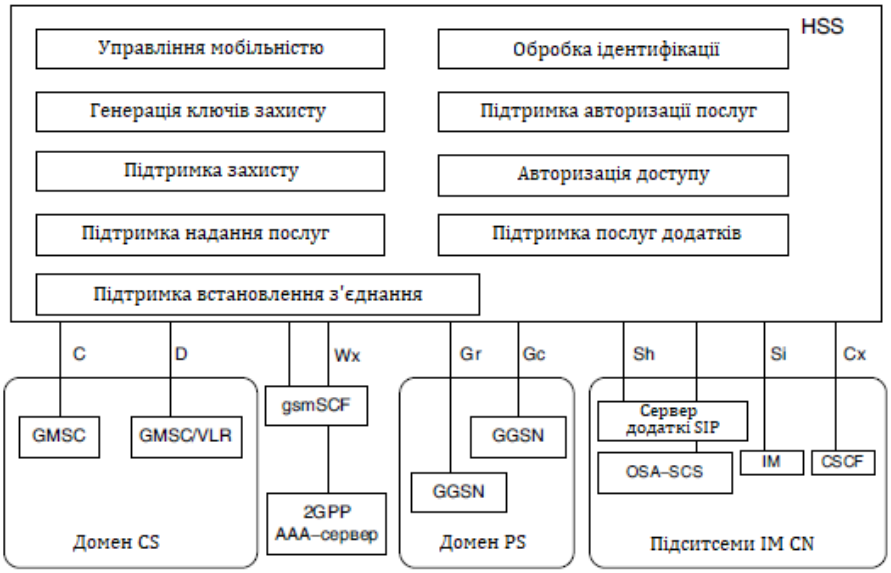

Рисунок 1.2.2.1 Логічні функції HSS

Мережа може містити більше одного HSS в тому випадку, якщо кількість абонентів занадто велика, щоб підтримуватися одним HSS. Така мережа, поряд з кількома HSS, повинна буде мати в своєму складі функцію SLF (Subscriber Location Function), що представляє собою просту базу даних, яка зберігає дані і відповідність інформації HSS адресами користувачів. [4]

3) Функція Policy Decision Function (PDF)

Функція Policy Decision Function (PDF) інколи інтегрується з P-CSCF, але може бути реалізована окремо. Ця функція відповідає за вироблення політики на підставі інформації про характер сеансу і про переданий трафік (транспортні адреси, ширина смуги і т.д.), отриманої від P-CSCF. На базі цієї інформації PDF

приймає рішення про авторизацію запитів від GGSN (Gateway GPRS Service Node) і виробляє повторну авторизацію при зміні параметрів сеансу, а також може заборонити передачу певного трафіку або організацію сеансів деяких типів.[4]

4) Сервери додатків

Сервери додатків AS (Application Server), по суті, не є елементами IMS, а працюють як би поверх неї, надаючи послуги в мережах, побудованих згідно IMS-архітектури. Сервери додатків взаємодіють з функцією S-CSCF по протоколу SIP. Основними функціями серверів додатків є обслуговування і модифікація SIP-сеансу, створення SIP-запитів, передача тарифікаційної інформації засобам нарахування плати за послуги.

Сервери додатків можуть бути дуже різними, але в IMS прийнято виділяти три типи серверів: SIP AS, OSA-SCS (Open Service Access - Service Capability Server), IM-SSF (IP Multimedia Service Switching Function):

• SIP AS - класичний сервер додатків, що надає мультимедійні послуги на базі протоколу SIP;

• OSA-SCS надає інтерфейс до сервера додатків OSA і функціонує як сервер додатків з боку S-CSCF і як інтерфейс між сервером додатків OSA і OSA API - з іншого боку;

• IM-SSF дозволяє використовувати в IMS послуги CAMEL (Customized Applications for Mobile Network Enhanced Logic), розроблені для GSM мереж, а також дозволяє керуючій функції gsmSCF (GSM Service Control Function) керувати IMS-сеансом. З боку S-CSCF сервер IM-SSF функціонує як сервер додатків, а з іншого боку - як функція SSF (Service Switching Function), що взаємодіє з gsmSCF по протоколу CAP (CAMEL Application Part).

Сервери додатків можуть перебувати або в домашній, або в будь-якій інший мережі, з якою у провайдера є сервісна угода. Але якщо сервер додатків знаходиться у зовнішній мережі, він не може мати інтерфейс з HSS. [4]

5) MRF (Media Resource Function)

MRF є джерелом медіа-інформації в домашній мережі і дозволяє відтворювати різні оголошення, змішувати медіа-потоки, транскодувати бітові потоки кодеків, отримувати статистичні дані та аналізувати медіа-інформацію. Функція MRF ділиться на дві частини:

• MRFC - Media Resource Function Controller

• MRFP - Media Resource Function Processor

MRFC знаходиться на сигнальному рівні і взаємодіє з S-CSCF по протоколу SIP. Використовуючи отримані інструкції, MRFC управляє по протоколу MEGACO / H.248 процесором MRFP, що знаходяться на рівні передачі даних, а той виконує всі маніпуляції з медіа-інформацією. Сама MRF завжди знаходиться в домашній мережі. [4]

6) BGCF (Breakout Gateway Control Function)

Breakout Gateway Control Function - це SIP-сервер, здатний виконувати маршрутизацію викликів на основі телефонних номерів. BGCF використовується тільки в тих випадках, коли сеанс ініціюється IMS-терміналом, а адресатом є абонент мережі з комутацією каналів (наприклад, ТфЗК або мобільної мережі 2G).

Основними завданнями BGCF є вибір тієї IMS-мережі, в якій повинна відбуватися взаємодія з мережею комутації каналів, або вибір відповідного PSTN (Public Switched Telephone Network)/CS шлюзу, якщо ця взаємодія має відбуватися в мережі, де знаходиться сам сервер BGCF. У першому випадку BGCF переводить сеанс до BGCF обраної мережі, а в другому - до обраного PSTN/CS шлюзу. [4]

7) Шлюз PSTN/CS

Шлюз PSTN/CS Gateway (рисунок 1.2.2.2) підтримує взаємодію IMS-мережі з ТфЗК і дозволяє встановлювати з'єднання між користувачами цих мереж. Шлюз PSTN/CS має розподілену структуру, характерну для архітектури Softswitch:

• SG (SGW) - Signaling Gateway;

• MGCF - Media Gateway Control Function;

• MGW - Media Gateway;

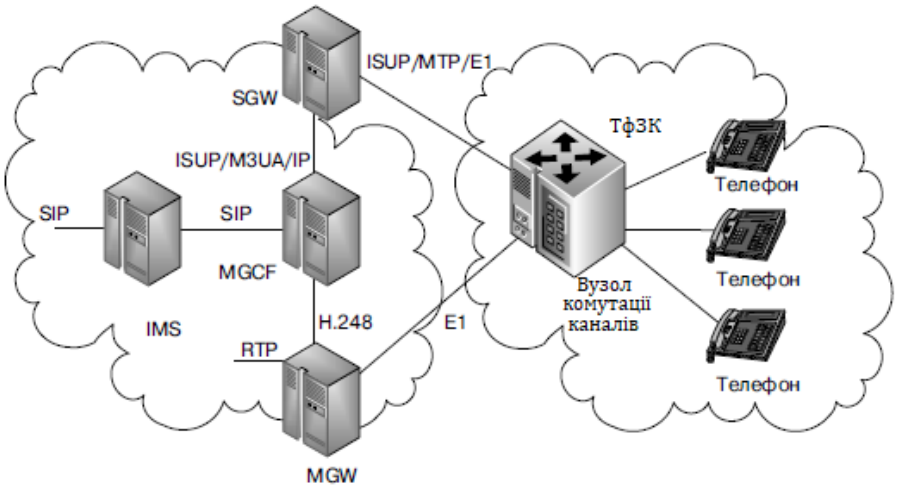

Рисунок 1.2.2.2 Шлюз PSTN/CS

Шлюз сигналізації SG є інтерфейсом для зв'язку з рівнем сигналізації в мережі комутації каналів, проводить перетворення нижніх протокольних рівнів систем сигналізації для двостороннього сигнального обміну між мережею IP і мережею ТфЗК. При цьому SG не виконує жодних перетворень над повідомленнями прикладного рівня.

Функція управління медіа-шлюзом MGCF - центральна частина розподіленого шлюзу PSTN/CS. Вона перетворює повідомлення ISUP (ISDN User Part) і BICC (Bearer independent call control protocol), які надходять з боку ТфЗК, в повідомлення SIP, які IMS використовує для управління сеансом зв'язку через границю між мережами (Рисунок 1.2.2.3). Крім меппінга сигнальних протоколів, MGCF управляє по протоколу MEGACO (Media Gateway Control Protocol)/H.248 ресурсами медіашлюзу, який бере участь в створенні з'єднання.

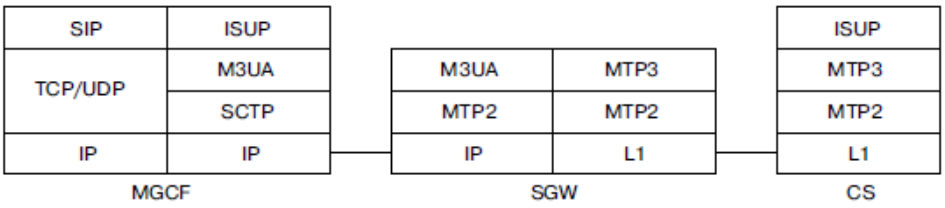

Рисунок 1.2.2.3 Управління сеансом зв’язку

Транспортний шлюз MGW з'єднує мережу IP з мережею комутації каналів на рівні передачі трафіку, виконуючи двостороннє перетворення призначеного для користувача трафіку, що проходить через кордон між мережами. З боку IMS-мережі шлюз MGW веде передачу і прийом даних у вигляді RTP-пакетів, а з боку мережі з комутацією каналів реалізує стандартний TDM-інтерфейс. Крім того, MGW може виконувати транскодування інформації, якщо в IMS і в мережі комутації каналів використовуються різні мовні кодеки (зазвичай в IMS використовується AMR (Adaptive Multi Rate), а в ТМЗК G.711).

1.3 Мережі мобільного зв’язку 4G LTE

Технологія 4G Long Term Evolution (LTE), розроблена Third Generation Partnership Project (3GPP), є терміном, який використовується для означення високоефективної технології четвертого покоління (4G) та мережевої архітектури. Основним завданням LTE є збільшення пропускної спроможності мережі за рахунок забезпечення більшої швидкості передачі трафіку з меншою затримкою пакетів. Це дозволить розширити сферу діяльності бездротового зв’язку, що в свою чергу забезпечить підтримку мультимедійних додатків та реалізацію справжньої бездротової широкосмуговості.

4G LTE засновано на абсолютно новій концепції в мобільних мережах, яка спрощує архітектуру, в той же час, зі збільшенням зв’язності її елементів. Реалізація LTE створює більш ефективну та гнучку мережу, просту в розгортанні та експлуатації.

4G LTE забезпечує більші швидкості передачі даних при менших витратах, покращену спектральну ефективність із зменшеною затримкою та гнучку пропускну здатність каналу. Для досягнення цих цілей специфікації LTE працюють з мережевими елементами (NE) через спеціальні інтерфейси. Найважливіше і головне для LTE - це простота роботи і взаємодіяти зі всіма іншими IP мережами лише за допомогою одного, використовуючи при цьому

плоску архітектуру на основі пакетної передачі, що виключає окремі контролери.

Впровадження LTE суттєво впливає на всі аспекти проектування та експлуатації мережі. Операторам мережі необхідно розробити всебічну мережеву архітектуру (E2E) від мобільного сайту до ядра мережі, яка може працювати використовуючи всі переваги нового стандарту LTE, одночасно залишаючись здатними оптимізувати застарілі 2G та 3G мережі, при цьому продовжуючи надання послуг користувачам. Мережі повинні підтримувати весь спектр технологій, послуг, розгортання та міграційних шляхів. [5]

Отже, виділимо основні вимоги до даних мереж:

- Швидкість передачі даних вище 100 Мбіт / сек.;
- Високий рівень безпеки системи;
- Висока енергоефективність;
- Низькі затримки в роботі системи;
- Сумісність зі стандартами другого і третього поколінь.

1.3.1 Архітектура мереж 4G LTE

Архітектура мережі LTE розроблена таким чином, щоб забезпечити підтримку пакетного трафіку з так званої "гладкою" ("безшовною", seamless) мобільністю, мінімальними затримками доставки пакетів і високими показниками якості обслуговування. Мобільність як функція мережі забезпечується двома її видами: дискретної мобільністю (роумінгом) і безперервної мобільністю (хендовером). Оскільки мережі LTE повинні підтримувати процедури роумінгу і хендовера з усіма існуючими мережами, для LTE-абонентів (терміналів) має забезпечуватися повсюдне покриття послуг бездротового широкосмугового доступу.

При розробці архітектури мережі 4G LTE були прийняті до уваги наступні загальні принципи.

- Логічно розділені транспортні (під)мережі передачі користувацьких даних і службової інформації;
- Мережа радіодоступу і базова пакетна мережа повністю звільнені від транспортних функцій. Схеми адресації, використовувані в цих мережах, не повинні бути пов'язані зі схемами адресації, що використовуються при реалізації транспортних функцій. Той факт, що деякі функції мережі радіодоступу або базової пакетної мережі фізично реалізовані в тому ж обладнанні, що і деякі транспортні функції, не говорить про те, що транспортні функції є частиною зазначених мереж;
- Управління мобільністю абонентів і/або призначених для користувача терміналів повністю покладено на мережу радіодоступу;
- Функціональний розподіл інтерфейсів мережі радіодоступу повинен мати кілька можливих опцій;
- Інтерфейси повинні базуватися на логічній моделі блоку, керованого даними інтерфейсом;
- Один фізичний елемент мережі може реалізаційно мати в собі кілька логічних блоків.

В мережах 4G LTE пакетна передача даних дозволяє забезпечити роботу всіх послуг, включаючи передачу призначеного для користувача голосового трафіку. На відміну від більшості мереж попередніх поколінь, в яких спостерігається достатньо висока різнотипність і ієрархічність мережевих вузлів (так звана розподілена мережева відповідальність), архітектуру мереж LTE можна назвати "плоскою", оскільки практично вся мережева взаємодія відбувається між двома вузлами: базової станцією (БС), яка в технічних специфікаціях називається B-вузлом (Node-B, eNB) і блоком управління мобільністю MME (Mobility Management Entity), реалізаційна, як правило, включає і мережевий шлюз GW (Gateway), тобто мають місце комбіновані блоки MME/GW. [6]

Архітектуру мережі 4G LTE представлено на рисунку 1.3.1.1.

Структура мережі 4G LTE сильно відрізняється від структури мереж стандартів 2G і 3G. Істотні зміни зазнала підсистема базових станцій і підсистема комутації. Була змінена технологія передачі даних між обладнанням користувача та базовою станцією. Також піддалися зміні і протоколи передачі даних між мережевими елементами. Тобто, як було вказано раніше в даному пункті, вся інформація (голос, дані) передається у вигляді пакетів. Таким чином, вже немає поділу на частини обробки або тільки голосової інформації, або тільки пакетних даних.

Виділяються наступні основні елементи мережі стандарту 4G LTE:

1) Serving SAE Gateway або просто Serving Gateway - обслуговуючий шлюз мережі LTE. Призначений для обробки і маршрутизації пакетних даних, що надходять з/в підсистему базових станцій. SGW має пряме сполучення з мережами другого і третього поколінь того ж оператора, що спрощує передачу з'єднання в/з них з причин погіршення зони покриття, перевантажень та інших параметрів. У SGW немає функції комутації каналів для голосових з'єднань, тому що в LTE вся інформація, включаючи голос комутується і передається за допомогою пакетів.

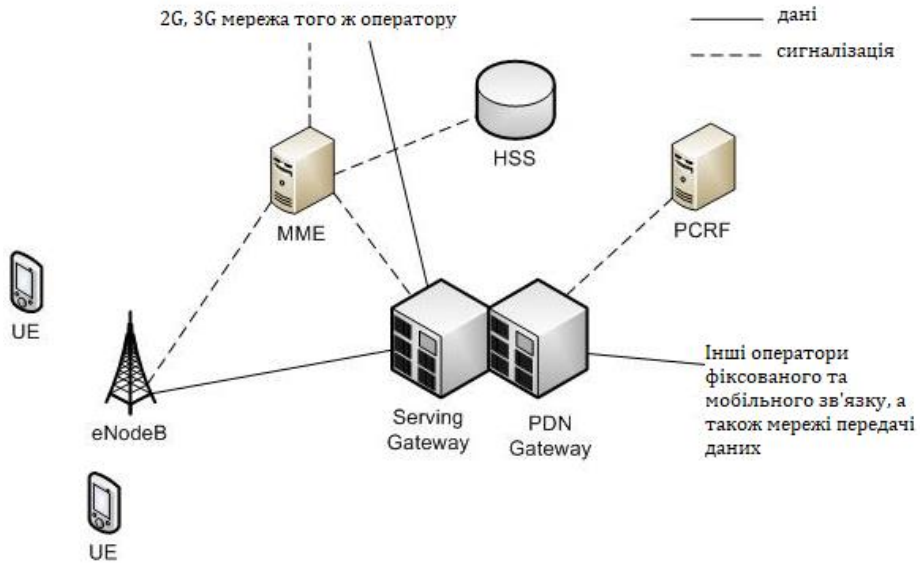

Рисунок 1.3.1.1 Узагальнена архітектура мережі 4G LTE

2) Public Data Network SAE Gateway або просто PDN Gateway - шлюз до мереж передачі даних інших операторів для мережі 4G LTE. Основне завдання PGW полягає в маршрутизації трафіку мережі LTE до інших мереж передачі даних, таких як Інтернет, а також мереж GSM, UMTS (Universal Mobile Telecommunications System).

3) Mobility Management Entity (MME) - вузол управління мобільністю мережі стільникового зв'язку стандарту 4G LTE. Призначений для обробки сигналізації, переважно пов'язаної з управлінням мобільністю абонентів в мережі.

4) Home Subscriber Server (HSS) - сервер абонентських даних мережі стільникового зв'язку стандарту 4G LTE. Являє собою велику базу даних і призначений для зберігання даних про абонентів. Крім того, HSS генерує дані, необхідні для здійснення процедур шифрування, аутентифікації і т.п. Мережа LTE може включати один або декілька HSS. Кількість HSS залежить від географічної структури мережі і числа абонентів.

5) Policy and Charging Rules Function (PCRF) - елемент мережі стільникового зв'язку стандарту 4G LTE, який відповідає за управління нарахуванням плати за надані послуги зв'язку, а також за якість з'єднань відповідно до заданих конкретному абоненту характеристиками.[7]

Варто відзначити, що для того, щоб дані могли бути транспортовані через інтерфейс радіо LTE, залучаються інформаційні канали. Вони використовуються для того, щоб виділяти різні типи даних і дозволити їм транспортуватися через мережу доступу більш ефективно. Використання декількох каналів забезпечує інтерфейс більш високого рівня в рамках протоколу LTE і включає більш чітку і визначену сегрегацію даних.

Є три категорії, в які можуть бути згруповані різні канали передачі даних:

1) Логічні канали – надають послуги середнього рівня управління доступом MAC (Medium Access Control) в межах структури протоколу LTE. Логічні канали по типу переданої інформації діляться на логічні канали управління і логічні канали трафіку. Логічні канали управління

використовуються для передачі різних сигнальних та інформаційних повідомлень. На логічних каналах трафіку передають призначені для користувачів дані.

2) Транспортні канали - транспортні канали фізичного рівня забезпечують передачу інформації в MAC і вище. Інформацію логічних каналів після обробки на RLC (Radio Link Control)/MAC рівнях розміщують в транспортних каналах для подальшої передачі по радіоінтерфейсу в фізичних каналах. Транспортний канал визначає як і з якими характеристиками відбувається передача інформації по радіоінтерфейсу. Інформаційні повідомлення на транспортному рівні розбивають на транспортні блоки. У кожному часовому інтервалі передачі TTI (Transmission Time Interval) по радіоінтерфейсу передають хоча б один транспортний блок. При використанні технології MIMO (Multiple Input Multiple Output) можлива передача до чотирьох блоків в одному TTI.

3) Фізичні канали - це канали передачі, які переносять призначені для користувача дані і повідомлення сигналізації (управління). Вони змінюються між вихідними і вхідними потоками, оскільки кожен з них має різні вимоги і діє по-своєму. [8]

1.3.2 Застосування концепції IMS в мережах 4G LTE

Як вже згадувалось раніше основною особливістю мобільних мереж четвертого покоління 4G LTE, на відміну від 2G, 3G є те, що передача будь-якого виду інформації в них відбувається за допомогою пакетних технологій. Так перед операторами, що планували перехід на мережі 4G LTE, постала проблема у зв’язності з мережами попередніх поколінь і забезпеченням передачі, насамперед, голосової інформації між різними користувачами, частина з яких є користувачами мереж з технологією комутації каналів.

Можливості передачі голосових даних в мережах четвертого покоління (4G) були досить обмежені. З розвитком технологій, послуга Voice over IP

прийшла в мобільні мережі 4G, і носить назву Voice over LTE (VoLTE) і зав'язана на підсистемі IMS (Рисунок 1.3.2.1). Саме концепція мультимедійної IP підсистеми дозволила створити необхідну зв’язність мереж різних поколінь і забезпечити роботу мережі 4G LTE на пакетній технології, тобто забезпечити перехід від CS (Chanel Switching – комутація каналів) до PS (Packet Switching - комутація пакетів). Архітектуру мережі оператора зв’язку, що працює з мережами різних поколінь, з впровадженою в неї підсистемою IMS зображено на рисунку 1.3.2.2.

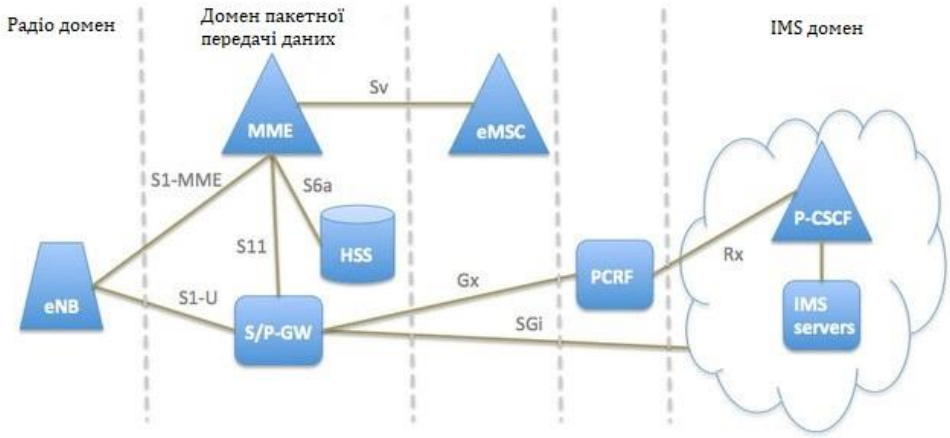

Рисунок 1.3.2.1 Структура мережі 4G з впровадженою в неї підсистемою IMS

Саме ж поняття IMS з'явилося ще в 2000 році, коли консорціум 3GPP почав роботу над стандартом Release 2000. В результаті IMS зустрічається вже починаючи з версії стандарту 3GPP Release 5 (березень 2002 року), хоча архітектура з усім функціоналом була остаточно приведена до нормального виду тільки до початку 2004 року. У більш пізніх редакціях (Release 7) додали підтримку різних стандартів. Починаючи з Release 8 (2008 рік) концепцію IMS можна розглядати як робочу модель для передачі голосових даних в мережах 4G LTE, хоча тільки в наступному Release 9 (2010 рік) додали підтримку екстрених викликів, Location Control Services (LCS) та інші поліпшення. [9]

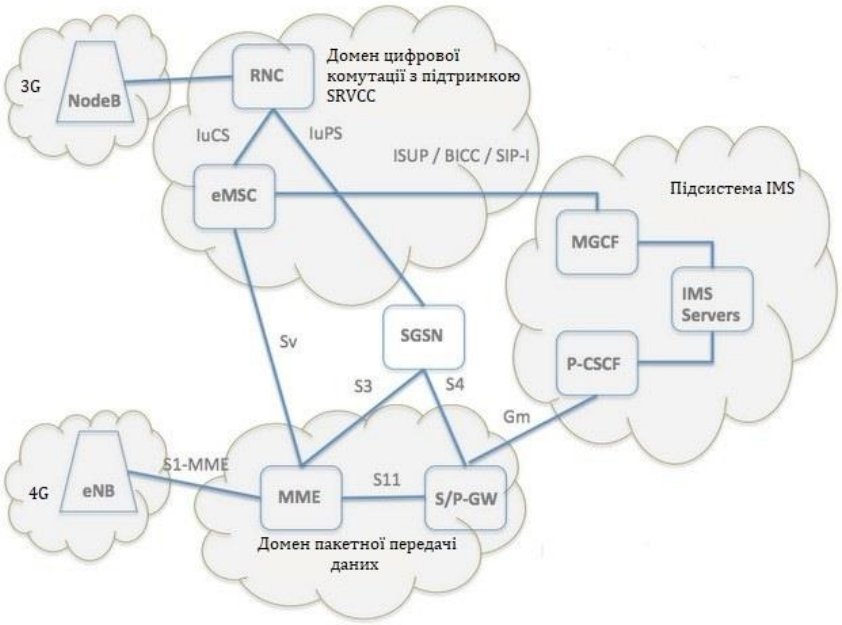

Рисунок 1.3.2.2 Мережева архітектура мережі оператора зв'язку з впровадженою підсистемою IMS

1.3.3 Поява нових послуг в рамках мереж 4G LTE з впровадженням в неї підсистеми IMS

При запровадженні концепції IMS в мережі 4G LTE виникає можливість до надання користувачам великої кількості нових послуг. Даний набір послуг отримав назву MMTel (IMS Multimedia Telephony) і був вперше стандартизований консорціумом 3GPP в їхньому Release 7. 3GPP Release 8 продовжив стандартизацію MMTel сервісів, але не забезпечив можливість для кінцевого користувача узгоджено маніпулювати налаштуваннями послуг для мереж 2G / 3G і MMTel. Остаточне рішення було запропоновано в 3GPP Release 9. Варто зазначити, що організація TISPAN (Telecommunications and Internet converged Services and Protocols for Advanced Networking) також проводила свою стандартизацію надання додаткових послуг в своїх Release 1 TISPAN і Release 2 TISPAN, але з виходом останнього ними було прийнято рішення передати всі питання щодо стандартизації до 3GPP в рамках загальної стандартизації мереж IMS.

MMTel (IMS Multimedia Telephony) – представляє собою стандартизований набір мультимедійних сервісів для абонентів мобільних і фіксованих мереж. MMTel дозволяє встановлювати з'єднання між користувачами, а також надавати користувачам додаткові послуги (supplementary services).

Набір мультимедійних послуг MMTel складається з двох частин:

- Базове з'єднання (basic communication);
- Опціональні додаткові сервіси (optional supplementary services). Для підтримки даних послуг на мережі повинен бути встановлений сервер телефонних додатків TAS (Telephony Application Server), який представляє з себе SIP-сервер, що забезпечує підтримку додаткових сервісів MMTel. [10]

1) Базове з’єднання включає в собі наступні стандартизовані послуги:

- Голосовий зв'язок (Speech);
- Відео зв'язок (Video);
- Текстова зв'язок (Text);
- Передача факсимільних повідомлень (Fax);
- Передача файлів (File Transfer);
- Спільне використання відео, аудіо, зображення, файлів (Video sharing, Audio sharing, Image sharing, File sharing).

2) Опціональні додаткові сервіси включають в собі наступні стандартизовані послуги:

- Originating Identification Presentation (OIP). Послуга надає користувачеві можливість отримання довіреної (тобто наданої мережі) ідентифікаційної інформації абоненту, що його викликає ;
- Originating Identification Restriction (OIR). Послуга надає користувачеві можливість приховати свою індентифікаційну інформацію для інших абонентів;

- Terminating Identification Restriction (TIR). Послуга надає користувачеві можливість абоненту, якого будуть викликати, приховати свій ідентифікатор;
- Malicious Communication IDentification (MCID). Послуга надає користувачеві можливість ідентифікувати зловмисний виклик;
- Anonymous Communication Rejection (ACR). Послуга надає користувачеві можливість блокувати виклики анонімних користувачів;
- Communication DIVersion (CDIV). Послуга надає користувачеві можливість перенаправляти направлені на нього виклики на іншого користувача;
- Communication Waiting (CW). Послуга надає користувачеві можливість отримувати сповіщення про вхідні виклики при вже активній сессії;
- Communication HOLD (HOLD). Послуга надає користувачеві можливість призупинити роботу активної мультимедійної сессії, а згодом її відновити;
- Communication Barring (CB). Послуга надає користувачеві можливість вибірково блокувати встановлення певних з'єднань;
- Completion of Communications to Busy Subscriber (CCBS). Послуга надає користувачеві, який викликає іншого користувача, що в той же час має активну сесію, можливість отримати сповіщення про завершення сеансу користувача, що був зайнятий;
- CONFerence (CONF). Послуга надає користувачеві можливість організувати сеанс з двома або більше іншими користувачами;
- Advice Of Charge (AOC). Послуга надає користувачеві можливість отримувати сповіщення про вартість використовуваних ним послуг;
- Explicit Communication Transfer (ECT). Послуга надає користувачеві можливість перенаправити його активну сесію на іншого користувача;
- Closed User Group (CUG). Послуга надає користувачеві можливість до формування групового сеансу зв'язку серед користувачів, частина з яких не має можливість здійснювати або приймати виклики;

- Flexible Alerting (FA). Послуга надає користувачами, які вже знаходиться в групі, можливість здійснити з’єднання з користувачами іншої групи (виклик група до групи);
- Completion of Communications on No Reply (CCNR). Послуга надає користувачеві можливість здійснювати автоматичний дзвінок до іншого користувача, який не відповів на попередній дзвінок, якщо інший користувач проявить якусь активність у мережі; [11]

Перерахуємо переваги які матимуть, як оператори зв’язку, так і користувачі, при введенні в мережу набору мультимедійних послуг MMTel.

Переваги для оператора:

1) Він надає користувачам повне та безпроблемне обслуговування послуг - незалежно від того, чи є їх пристрій мобільним чи фіксованим;

2) Користувачі отримують доступ до кількох мультимедійних функцій, включаючи відео, чат та обмін зображеннями;

3) Характеристики MMTel були розроблені з метою заміни всіх поточних рішень з фіксованої та мобільної телефонії. Це означає, що, обираючи MMTel, оператори можуть консолідувати свої мережі та зменшити CAPEX (Capital Expenditure) та OPEX (Operating expense);

4) Технології мобільного доступу, такі як LTE та WiMAX (Worldwide Interoperability for Microwave Access), не підтримують технологію голосового зв’язку через пакетні мережі;

5) Глобальні угоди про взаємозв'язок щодо послуг між операторами є ключовими для телекомунікаційної галузі. Стандартизована HHI дозволяє операторам взаємодіяти один з одним, створюючи потенціал для справді глобального впровадження на масовий ринок;

6) Оскільки MMTel - стандарт, виготовлений для масового ринку, він підтримуватиме розробку недорогих пристроїв. Оператори, які обирають стандарт MMTel для збереження своїх позицій у ланцюзі вартості, можуть використовувати IMS та користь від швидкої та інноваційної розробки додатків. Можливість здійснити телефонний дзвінок або надіслати SMS кому-

небудь і знати, що зв’язок працюватиме між багатьма операторами та постачальниками послуг - це потужна концепція, яку можна досягти лише за допомогою глобальних стандартів. Те саме стосується всіх нових послуг, включених стандартом MMTel. [11]

Переваги для користувачів:

Основні комунікаційні можливості MMTel дозволяють за допомогою одного сеансу SIP контролювати передачу медіа даних. Таким чином, два або більше користувачів можуть спілкуватися в режимі реального часу, використовуючи різні медіа компоненти, включаючи:

1) Голос у режимі реального часу, можливість використання застарілих кодеків, таких як адаптивний багатосторонній AMR або широкосмуговий кодек, такий як AMR-WB, для подальшого підвищення якості зв'язку;

2) Голосова синхронізована передача відео в режимі реального часу. Відеопотік може бути простим, щоб реалізувати послугу обміну відео або повний дуплекс;

3) Реалізувати послугу відеотелефонії. Найсучасніші відео кодеки, такі як H.264, можуть бути налаштовані для використання з високою швидкістю, щоб забезпечити значно більш високу якість відео, ніж користувачі, які звикли користуватися застарілими послугами відеотелефонії;

4) Передача тексту в режимі реального часу (символ на персонажа), який можна використовувати для створення “teletypewriter”, телекомунікаційного пристрою для людей з вадами слуху;

5) Передача тексту в режимі реального часу (одне повідомлення за раз), яке використовується для реалізації послуги чату на базі сеансу;

6) Обмін зображеннями, відеокліпами та аудіокліпами, що дозволяє користувачам ділитися файлами, які відображаються або відтворюються на приймальному терміналі безпосередньо при отриманні, використовуючи визначені формати файлів;

7) Передача файлів для обміну файлами будь-якого типу. Послуга сама по собі є доступною, оскільки вона базується на транспорті IP та сеансі SIP/SDP (Session Description Protocol). Користувачі можуть додавати та видаляти різні типи медіа даних, не припиняючи або перезавантажуючи сеанс. Таким чином, стандарт MMTel дозволяє за один сеанс SIP контролювати майже всі сервіси. У стандарті чітко визначені специфікації взаємозв'язку оператор - оператор. Це означає, що користувачі, що належать до різних операторів, можуть спілкуватися один з одним за допомогою усіх доступних мультимедійних послуг. [11]

Підсумовуючи наведену вище інформацію можна стверджувати, що послуги, які надаються користувачам інфокомунікаційного оператора є досить обширними. Хоча багато з них і повторюють ті послуги, що надаються OTT (Over-The-Top Service) сервісами, але не варто забувати і те, що кінцевому користувачу не потрібно завантажувати додаткових додатків, мігрувати поміж ними в пошуках найкращого сервісу, він може сплачувати лише за ті послуги, що йому справді потрібні і для цього ж користувача буде надаватись найкраща якість сервісу, яка доступна в даний момент у географічний точці де він знаходиться, що є одним із принципів побудови IMS мереж.

1.4 Передумови переходу мультимедійної IP підсистеми до «хмарних» технологій

Зростаюча з кожним днем кількість обороту трафіку у світі і як наслідок широка популярність IP мереж та обсяг мультимедійних послуг, що вони надають, створюють можливості і передумови до появи нових бізнес-моделей у всіх типах мереж доступу та підключених пристроїв. Наразі затрати операторів зв’язку на купівлю нового обладнання, а часто і різновендерного, створення нових точок доступу користувачів, побудову оптичних ліній зв’язку, а також підтримку у робочому стані всього перерахованого є дуже затратним для сучасних постачальників інфокомунікаційних послуг. Тому в сучасному світі

все більше компаній-постачальників послуг розглядають питання про перехід до віртуалізації мережевих функцій NFV (NFV), так вони шукають можливості монетизувати та реалізувати операційні переваги у віртуалізації та зменшити затрати на обслуговування своєї мережі.

Один з таких випадків обертається навколо мультимедійної IP підсистеми (IMS), мережевого рішення, що дозволяє постачальникам послуг пропонувати цілий спектр послуг, таких як голос споживачів через IP (VoIP), голос через LTE (VoLTE (Voice over LTE)) та голос через WiFi (VoWiFi (Voice over Wireless Fiber)). Таким чином, IMS став життєво важливим елементом майбутнього мобільних мереж, якщо оператор хоче бути насправді конкурентоспроможним. Керовані попитом споживачів на мультимедійні додатки, стрімінгові платформи музики та відео, постачальники послуг прискорюють темпи розвитку своїх мереж та створюю каркас для мереж, що будуть затребувані у майбутньому. Віртуалізація IMS розглядається як ключовий крок у цьому процесі.

Аргумент для переходу до віртуалізованого ядра IMS (vIMS (Virtual IMS)) є переконливим, і постачальники послуг зараз розглядають конкретні пропозиції, які можна ефективно розгорнути на ядрі vIMS. VoLTE - одна з головних послуг IMS та один з найбільш вагомих доказів для віртуальної IMS. VoLTE дозволяє мережам 4G підтримувати голосові дзвінки, пропонуючи постачальникам послуг потенційно надійний потік доходу, щоб повернути свої інвестиції на модернізацію мереж до покоління 4G. VoLTE на vIMS розглядається як диференціатор для сервіс провайдерів, що пропонує розширені голосові та відео послуги споживачам та розширену IP-телефонію для корпоративних та житлових клієнтів.

То які ж виклики постають перед провайдерами послуг?

Оскільки конкуренти традиційних мереж, OTT сервіси, постійно знижують доходи постачальників послуг через пропозиції різного роду сервісів, такі як месенджери та сервіси VoIP, постачальники послуг прагнуть зберегти переваги, у всіх можливих аспектах. Однією з таких переваг є доступ до аналізу

даних про використання трафіку абонентів, що надає сервіс провайдерам можливість налаштувати пакети послуг для залучення різного роду клієнтів.

Використовуючи віртуалізовану архітектуру NFV, постачальники послуг отримують переваги швидкості та гнучкості у тому, щоб швидше поставляти послуги на ринок. Використовуючи дані по використанню трафіка за послугами, що вже доступні в мережі, вони також можуть пропонувати більш диференційовані та необхідні користувачам послуги.

Постачальники послуг постійно шукають нові потоки доходів для підтримки інвестицій у нові технології. Їм необхідно підтримувати середовища, що беруться в оренду, з метою включення інтегрованих мультимедійних послуг у підключені пристрої та оптового продажу голосових послуг підприємствам або стороннім постачальникам послуг IT. Можливість розширення та модифікації основних функцій IMS для включення нових бізнес-моделей дає операторам гнучкість у наданні послуг при різних SLA (Service-level agreement) у відповідності з потребами клієнта.

Нові бізнес моделі, що виникають при реалізації vIMS (Virtual IMS):

- Побутові послуги - нові фіксовані пропозиції та постійне розширення послуг VoLTE, доповнені мобільною інтероперабельністю VoWiFi;
- Підприємницькі послуги - нові моделі доходів та пакетне обслуговування послуг, таких як IMS хостинг, IP-PBX (Private Branch eXchange), vSBC (Virtual session border controller)/SIP магістралі, веб-конференції, відеотелефонія, RCS (Revision Control System), безкоштовні послуги тощо;
- Віртуальні керовані сервіси - спільні або розділені мережеві додатки IMS у централізованій або крайній «хмарі», запускаються за потреби;
- Інтегрована комунікація - багата передача голосових, відео, текстових комунікаційних пропозицій через будь-яку мережу доступу та безліч пристроїв, наприклад - смартфони, планшети, фіксовані ПК, ноутбуки тощо.

Нові можливості для постачальників послуг, що створюються за допомогою vIMS:

• Час на ринок (Time to market) - Постачальники послуг можуть швидко виводити на ринок нові та диференційовані послуги передачі інформації, а також динамічно масштабувати їх відповідно до статистики використання, що ведеться в режимі реального часу та характеристиками мережі;

• Послуги з доданою вартістю (Value-added services) - віртуалізація ядра IMS надає постачальникам послуг можливість вирішити швидко зростаючу потребу в розширених послугах зв'язку через мобільні та фіксовані мережі. Розгортання таких послуг, як VoLTE та інтегровані мультимедійні послуги, приносять інновації, що потрібні операторам та клієнтам мережі;

• Якість досвіду (Quality of experience) - Постачальники послуг можуть запропонувати високу якість обслуговування (QoS), додатково відрізняючись від інших голосових послуг OTT сервісів. Поділ мережі та розподіл політик забезпечують необхідну гарантію рівня обслуговування в усіх сферах обслуговування клієнтів;

• Покращене надання послуг (Enhanced service delivery) - Складності в існуючому середовищі для багатьох постачальників та операційних процесах подовжують можливості надання послуг та їх оновлення. NFV дозволяє постачальникам послуг автоматизувати та спрощувати розгортання мережі з повноцінними операціями життєвого циклу, еластичністю сервісу, інтелектуальним управлінням потенціалом та контролем сегментації політики.

• Готовність до хмар (Cloud ready) - vIMS забезпечує конвергенцію та впорядкування гібридних хмар, пропонуючи персоналізовані та контекстуалізовані огороджені сади та публічні хмарні послуги. Постачальники послуг можуть охопити дозрілу екосистему постачальників SaaS та інтегрувати найкращі рішення (веб-конференції, чат тощо) у свої пакетні пропозиції, не змінюючи свої основні пропозиції;

• Підвищена сумісність (Increased interoperability) - З ростом проникнення мереж SIP та IPX, віртуальна система IMS забезпечує більш економічну взаємодію під час роботи з застарілими технологіями. [12]

1.5 Висновки до розділу 1

У розділі №1 розглядається структура та роль мультисервісної IP підсистеми (IMS) як у сучасних інфокомунікаційних мережах, так і необхідність її застосування у майбутньому. Так, у пунктах даного розділу наведено інформацію про те як дана архітектура виникла в результаті еволюційного процесу мереж NGN, перейнявши від них ідею функціонування на основі технології з комутацією пакетів та забезпечення конвергентної роботи послуг, що в ній надаються.

Також, в рамках даного розділу наводиться інформація про функціонування сучасних мобільних мереж 4G LTE та їх безпосередній зв’язок з підсистемою IMS. Так з’ясовується, що конвергенція цих технологій дозволяє забезпечити надання широкого спектру нових послуг усім користувачам мережі, незалежно від місця їхнього розташування, або ж термінального обладнання яке вони використовують, при цьому забезпечуючи найбільшу з можливих якість сервісу відносно кожного з абонентів.

В результаті роботи над розділом, виясняється, що згідно з темпами росту пакетного трафіку в сучасному світі та тенденцією міграції користувачів між різними провайдерами, операторам інфокомунікаційних послуг доцільно розгортати свої мережі у хмарних середовищах. Віртуалізація їхніх мереж дозволить задовольняти потреби кожного з користувачів, шляхом гнучкого управління мережею та надання абонентам тих послуг, які їм потрібні, не переплачуючи. Таким чином оператор не тільки може спостерігати за трендами у потребах користувачів, швидко впроваджуючи нові послуги, а і, відповідно, зменшувати витрати на побудову та обслуговування своєї мережі.

РОЗДІЛ 2. ХМАРНІ ТЕХНОЛОГІЇ, ЯК ЗАСІБ ВІРТУАЛІЗАЦІЇ ІНФОКОМУНІКАЦІЙНИХ МЕРЕЖ ТА ПОСЛУГ

В сучасному світі та протягом попередніх років все більшу популярність і ефективність застосування завойовують хмарні технології і хмарні обчислення (англ. Cloud computing).

Хмарні обчислення досить загальний термін, який поєднує у особі декілька підходів та моделей з надання та управління IT (Information Technology) сервісами, тому на практиці кожен розуміє цей термін по різному. Хтось вважає хмарними обчисленнями хостинг віртуальних машин або колокейшн серверів лише за ознакою мережевого доступу до ресурсів, інші під хмарними обчисленнями розуміють такі користувальницькі сервіси, як Dropbox, Google Drive. Тобто більшість користувачів визначає хмарні обчислення лише за однією ознакою - мережевим доступом, але хмарні обчислення це набагато більш об'ємна сутність. Згідно з визначенням Національного інституту стандартів і технології (NIST (National Institute of Standards and Technology)) США, хмарні обчислення (від англ. Cloud Computing) — це модель забезпечення повсюдного та зручного доступу на вимогу, через мережу до спільного пулу обчислювальних ресурсів, що підлягають налаштуванню (наприклад, до комунікаційних мереж, серверів, засобів збереження даних, прикладних програм та сервісів), і які можуть бути оперативно надані та вивільнені з мінімальними управлінськими затратами та зверненнями до провайдера. Інакше кажучи хмарні обчислення являють собою концепцію надання IT ресурсів у вигляді послуг. [13]

Основною причиною впровадження таких технологій є економічний ефект, який надає їхнє використання. Всі проблеми, пов'язані з побудовою центрів обробки даних, купівлею серверного та мережевого обладнання, апаратних рішень, а також забезпеченням безперервності і працездатності, перекладаються на плечі провайдерів даних послуг. [14] А отже інфокомунікаційному провайдеру не потрібно купувати нове мережеве

обладнання, яке буде стрімко застарівати. Звичайно, процес «старіння» обладнання вендори уповільнюють за допомогою оновлення його програмного забезпечення, але з розширенням мережі оператору дані рішення, з часом, не будуть задовольняти всіх його потреб.

Але і звичайна оренда віддаленого апаратного забезпечення не буде оптимальним для всіх операторів послуг. Все більш тісний зв'язок процесів в сфері інфокомунікаційних технологій, стрімкий розвиток глобальних інформаційних і обчислювальних мереж, вимог клієнтів операторів зв’язку і ряд інших чинників ведуть до зміни фундаментальних парадигм обробки інформації внаслідок необхідності підтримки і розвитку розподілених інформаційно-обчислювальних ресурсів. Таким чином рішення хмарних обчислень, тобто надання користувачеві хмарних технологій абстрактних обчислювальних потужностей, які фізично розподілені на багатьох віддалених пристроях, що в свою чергу утворюють певну «хмару» буде значно зручнішим за попередні рішення, адже він платить не за покупку або користування конкретними апаратними ресурсами, а лише за безпосередньо виконані для нього розрахунки.

Так з точки зору постачальника, завдяки об'єднанню ресурсів і непостійного характеру споживання з боку споживачів, хмарні обчислення дозволяють економити на масштабах, використовуючи менші апаратні ресурси, ніж були потрібні б при виділених апаратних потужностях для кожного споживача, а за рахунок автоматизації процедур модифікації виділення ресурсів істотно знижуються витрати на абонентське обслуговування.

З точки зору споживача, ці характеристики дозволяють отримати послуги з високим рівнем доступності (англ. High availability) і низькими ризиками непрацездатності, забезпечити швидке масштабування обчислювальної системи завдяки еластичності без необхідності створення, обслуговування і модернізації власної апаратної інфраструктури. Зручність і універсальність доступу забезпечується широкою доступністю послуг і підтримкою різного класу

термінальних пристроїв (персональних комп'ютерів, мобільних телефонів, - планшетів). [13]

Таким чином Інформаційною Технологічною Лабораторією при Національному Інституту Стандартів та Технологій США (Information Technology Laboratory at the National Institute of Standards and Technology) було виділено основні характеристики хмарних обчислень:

- Самообслуговування за запитом. Споживач може в односторонньому порядку визначати і змінювати обчислювальні потреби, такі як серверний час та мережеве сховище даних, за необхідності автоматично, не вимагаючи взаємодії людини зі сторони постачальника послуг.
- Універсальний доступ по мережі, послуги доступні споживачам по мережі передачі даних незалежно від використовуваного термінального пристрою;
- Широкий доступ до мережі. Доступ до послуг здійснюється за допомогою стандартних механізмів незалежно від термінального обладнання користувача (наприклад, мобільних телефонів, планшетів, ноутбуків та робочих станцій).
- Об'єднання ресурсів. Обчислювальні ресурси постачальника об'єднані для обслуговування великої кількості споживачів, використовуючи модель множинної оренди. Різні фізичні та віртуальні ресурсами динамічно призначаються та перерозподіляються відповідно до потреб споживачів. У користувачів виникає відчуття у їхній незалежності від місцеположення в мережі, тому що, як правило, вони не мають контролю або знань щодо точного розташування обчислювальних ресурсів постачальника послуг, але той, в свою чергу, може бути в змозі визначити їхнє місцеположення на більш високому рівні абстракції (наприклад, країну, штат чи центр обробки даних).
- Еластичність, послуги можуть бути надані, розширені, звужені в будь-який момент часу, без додаткових витрат на взаємодію з постачальником, як правило, в автоматичному режимі;

- Еластичність. Обчислювальні можливості можуть еластично регулюватись, в деяких випадках автоматично, для швидкого масштабування пропорційно попиту. Можливості споживачів часто здаються необмеженими і можуть використовуватись в будь-якій кількості в будь-який час.
- Облік споживання, постачальник послуг автоматично обчислює спожиті ресурси на певному рівні абстракції (наприклад, обсяг збережених даних, пропускна спроможність, кількість користувачів, кількість транзакцій), і на основі цих даних оцінює обсяг наданих споживачам послуг.
- Вимірюваність послуг. Хмарні системи автоматично контролюють та оптимізують використання ресурсів, використовуючи можливість моніторингу на певних абстрактних рівнях відповідного типу послуги (наприклад зберігання, обробка даних, пропускна здатність та активні облікові записи користувачів). Використання ресурсів можна відстежувати та контролювати забезпечуючи прозорість як для постачальника, так і для споживача використовуваної послуги. [15]

Також виділяються основні моделі розгортання:

- Приватна хмара. Інфраструктура, призначена для використання однією організацією, що включає кілька споживачів (наприклад, підрозділів однієї організації), можливо також клієнтами і підрядниками даної організації. Приватна хмара може перебувати у власності, управлінні та експлуатації як самої організації, так і третьої сторони (або будь-якої їх комбінації), і воно може фізично існувати як всередині, так і поза юрисдикцією власника.
- Публічна хмара . Інфраструктура, призначена для вільного використання широкою публікою. Публічна хмара може перебувати у власності, управлінні та експлуатації комерційних, наукових та урядових організацій (або будь-якої їх комбінації). Публічна хмара фізично існує в юрисдикції власника — постачальника послуг.
- Гібридна хмара. Комбінація з двох або більше різних хмарних інфраструктур (приватних, публічних або суспільних), що залишаються унікальними об'єктами, але пов'язаних між собою стандартизованими або

приватними технологіями передачі даних і додатків (наприклад, короткочасне використання ресурсів публічних хмар для балансування навантаження між хмарами).

- Суспільна хмара. Вид інфраструктури, призначений для використання конкретною спільнотою споживачів з організацій, що мають спільні завдання (наприклад, місії, вимоги безпеки, політики, та відповідності різним вимогам). Громадська хмара може перебувати в кооперативній (спільній) власності, управлінні та експлуатації однієї або більше з організацій співтовариства або третьої сторони (або будь-якої їх комбінації), і вона може фізично існувати як всередині, так і поза юрисдикцією власника. [16]

2.1 Моделі обслуговування концепції хмарних обчислень

В сучасних інфокомунікаціях хмарні обчислення є перспективною та ефективною мережевий архітектурою надання багатокористувацького мережевого доступу до пулу колективних обчислювальних ресурсів - сервісів, додатків, сховищ даних, серверів, які абоненти в реальному часі в режимі «за вимогою» задіяють для отримання послуг і / або вирішення власних задач, а потім вивільняють після їх завершення. В рамках цієї архітектури користувачі в міру необхідності в автоматичному режимі використовують хмарні фізичні та віртуальні ресурси, стільникові сховища даних, серверний час, віртуальні машини і інше з використанням різних стандартних механізмів доступу, а провайдери, в свою чергу, об'єднують в пул, динамічно перерозподіляють, резервують, масштабують свої обчислювальні ресурси для обслуговування великого числа користувачів, контролюють завантаження своїх ресурсів, формують звіти про обсяг наданих послуг.

Можливі наступні три основні моделі обслуговування (Рисунок 2.1.1):

- Cloud SaaS (Software as a Service) - хмарне програмне забезпечення (ПО) як послуга;
- Cloud PaaS (Platform as a Service) - хмарна платформа як послуга;

- Cloud laaS (Infrastructure as a Service) - хмарна інфраструктура як послуга.

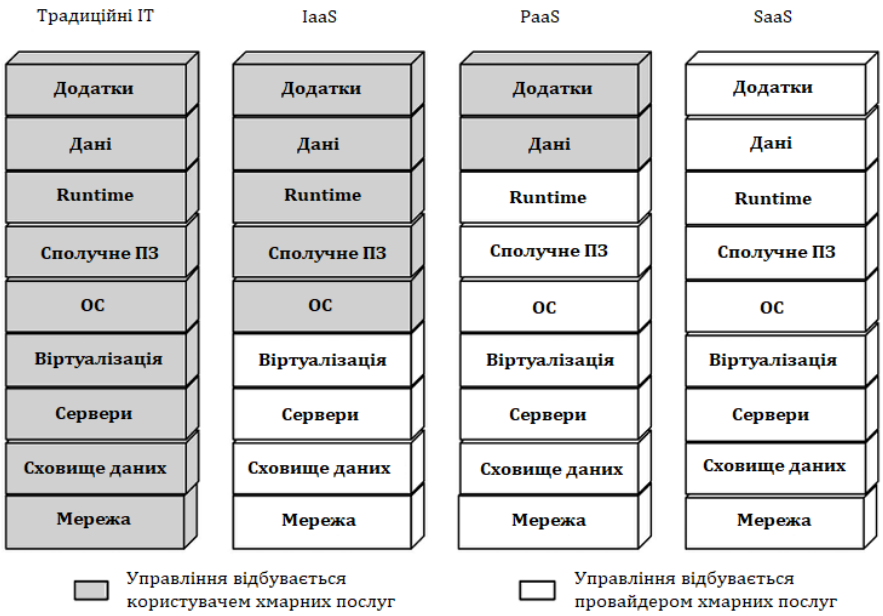

Рисунок 2.1.1 Моделі обслуговування хмарних сервісів

2.1.1 Програмне забезпечення як послуга

Модель SaaS хмарного програмного забезпечення як послуги надає користувачам готове прикладне ПЗ, розроблене, кероване і те яке обслуговується провайдером. Користувач не приймає участь у встановленні, оновленні і підтримці працездатності обладнання і працюючого на ньому програмного забезпечення, а лише віддалено користується ним, як правило, через веб-інтерфейс. або використовуючи тонкого клієнта.

Приклад наймасовішої і найчастіше використовуваної SaaS-послуги - сервіс електронної пошти Gmail. Більш спеціалізовані SaaS-сервіси - Dropbox, різні онлайнові органайзери, системи документообігу, спільного управління проектами і т. п. У малому і середньому бізнесі в режимі SaaS можуть застосовуватися системи CRM (Customer relationship management), білінгу і ERP (Enterprise Resource Planning System). [2]

2.1.2 Платформа як послуга

У моделі PaaS платформи як послуги користувачеві надається доступ до використання інформаційно-технологічної платформи хмарного провайдера, що включає в себе операційні системи, системи управління базами даних, засоби розробки, комунікаційне ПЗ. У цій моделі вся інформаційно-технологічна інфраструктура. включаючи обчислювальні мережі, сервери, системи зберігання, цілком управляється провайдером. Провайдером також визначається набір доступних для користувачів видів платформ і набір керованих параметрів платформ, а користувачеві надається можливість використовувати платформи, створювати їхні віртуальні екземпляри, встановлювати, розробляти, тестувати, експлуатувати на них прикладне ПЗ. При цьому динамічно змінюючи кількість споживаних обчислювальних ресурсів.

Відмінності моделі PaaS від SaaS ще і в тому. що PaaS надає засоби BI (Business Intelligence), які дозволяють користувачеві аналізувати дані, визначати тенденції і будувати прогнози для поліпшення планування і прийняття ефективних бізнесних рішень. PaaS надає також середовище для розробки, яке користувачі використовують для створення або настройки хмарних додатків. Хмарні функції масштабування, управління доступом, організація багатокористувацького режиму також вже включені в платформу і не вимагають додаткової роботи від користувача. [2]

Прекрасним прикладом PaaS є створена компанією Amazon служба AWS Elastic Beanstalk для розгортання і масштабування веб-додатків і сервісів, розроблених на Java. .NET, PHP, Python. Ruby та ін. Для написаного на цих мовах і завантаженого в систему коду Elastic Beanstalk автоматично виконає розгортання: виділяє ресурси, організовує балансування навантаження, автоматичне масштабування і моніторинг працездатності додатків. Користувач при цьому зберігає повний контроль над ресурсами і в будь-який час може отримати до них доступ.

2.1.3 Інфраструктура як послуга

Модель інфраструктури як послуги laaS передбачає можливість залучення користувачем хмарної інфраструктури для самостійного управління обчислювальними ресурсами, засобами обробки та зберігання даних, мережами та іншими фундаментальними обчислювальними ресурсами, операційними системами, платформним і прикладним програмним забезпеченням, користуватися деякими сервісами (Рисунок 2.1.3.1). [2]

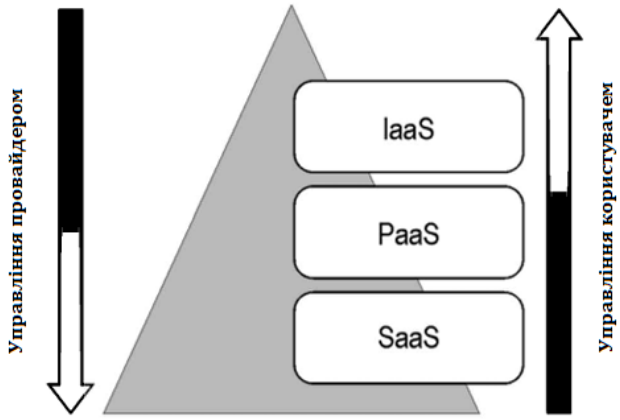

Рисунок 2.1.3.1 Відношення провайдер/користувач у хмарних сервісах

Управління ж основною фізичної і віртуальної інфраструктурою хмари, в тому числі мережі, серверів, типів використовуваних операційних систем, систем зберігання даних здійснюється хмарним провайдером. Світовими лідерами laaS є такі компанії, як Amazon, Microsoft. Google. IBM.

2.1.4 Все як сервіс

На додаток до SaaS, PaaS, laaS ряд інших хмарних сервісів передбачений в Рекомендації ITU-T Y.3500 "Cloud Computing - Overview and Vocabulary". Це хмарні сервіси - EaaS (Everything as a Service):

- CaaS (Communications as a Service) - відносяться відеоконференції, веб-конференції. обмін миттєвими повідомленнями і VoIP;
- Compute as a Service (CompaaS) - спрощений варіант laaS, що охоплює переважно хмарний обчислювальний ресурс;
- DSaaS (Data Storage as a Service), коли користувач орендує простір для зберігання даних, а також резервне копіювання даних і передачу даних через Інтернет.
- NaaS (Network as a Service). включає віртуальну приватну мережу VPN (Virtual Private Network), що налаштування маршрутизації, протоколи під LGPL, брандмауер безпеки, виявлення і запобігання вторгнень, моніторинг і антивірус.
- Database as a Service,
- Desktop as a Service і ін.

Останні розробки в області хмарних сервісів об'єднуються під назвою XaaS. де X має на увазі всеосяжний, а в реальності - просто різноманітний набір хмарних сервісів, який може бути наданий хмарним провайдером в розширення основної трійки SaaS, PaaS, laaS з деякими нюансами, але той же зміст вкладається і в терміни Anything as a Service і Everything as a Service.

Іншою послугою, що відноситься до управління інфокомунікаціям, є така система підтримки бізнесу, як сервіс BSSaaS (BSS-as-a-Service). який виводить в хмару найважливіший функціонал BSS. Сюди входять білінг, підтримка користувачів, взаєморозрахунки, онлайн-тарифікація. перед/постоплата і ін. Він же може бути поширений на завдання IoT (Internet of Things), розумного міста. Цілком прогнозуємо інтерес до зовсім нових спеціалізованих сервісів типу SECaaS (безпека як послуга), IDaaS (ідентифікація як послуга), DRaaS (Післяаварійне відновлення як послуга), BaaS (резервне копіювання як послуга), Big Data-as-a-Service та ін. [2]

2.2 Концепція віртуалізації мережних функцій - NFV (Network Functions Virtualization)

Суть концепції NFV полягає в поділі логіки функціонування мережевих елементів і устаткування, на якому вони виконуються за допомогою техніки віртуалізації фізичних ресурсів, тобто в перенесенні функціоналу мережевих елементів зі спеціалізованих програмно-апаратних телекомунікаційних пристроїв (BRAS (Broadband Remote Access Server), маршрутизатори , NAT (Network Address Translation), комутатори, брандмауери, DHCP (Dynamic Host Configuration Protocol), акселератори трафіку та ін.) на стандартні сервери (зокрема, ті ж x86) шляхом завантаження в них відповідного прикладного програмного забезпечення (Рисунок 2.2.1).

Привабливість цієї ідеї зростає все більше з кожним днем, адже кількість трафіку у мережах інфокомунікаційних операторів стрімко зростає. У цих умовах NFV дозволяє оператору позбавлятися від спеціалізованого, пропрієтарного і дорогого в обслуговуванні обладнання, замінюючи його віртуальними пристроями, тобто повністю програмними рішеннями на універсальних серверах. Ті чи інші мережеві функції реалізуються віртуально (програмуються) на універсальному, масовому, отже, недорогому обладнанні, наприклад на стандартних серверах виробництва HP, Juniper, Dell або IBM, завдяки чому оператори можуть значно знизити різноманітність використовуваного обладнання , а отже, оптимізувати витрати на його придбання і експлуатацію.

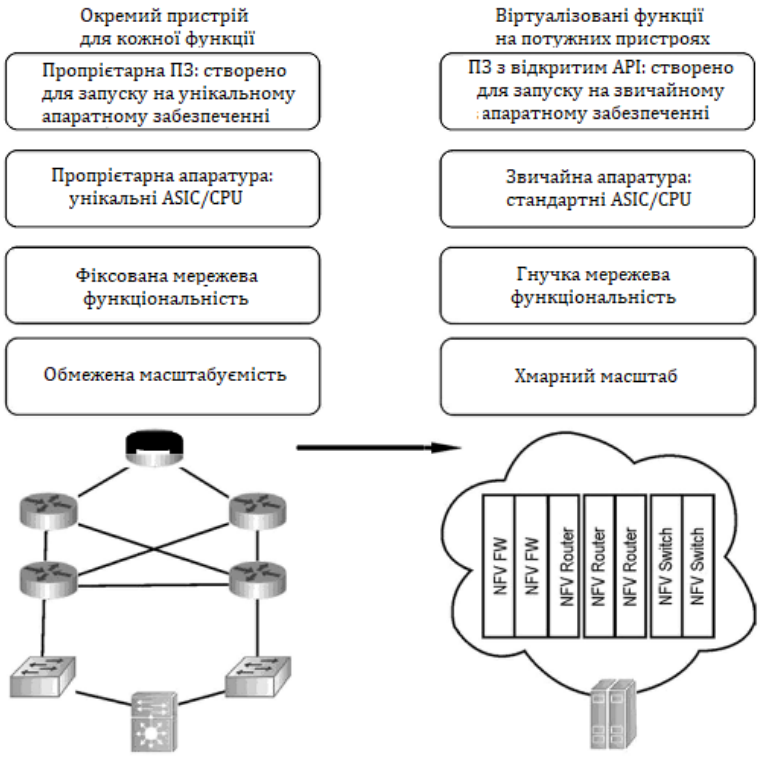

Рисунок 2.2.1 Перехід від традиційних мереж до NFV

2.2.1 Еталонна модель ETSI NFV MANO

Зрозуміло, ті переваги і нові можливості, які надає телекомунікаційним операторам концепція NFV, не даються даром. Потрібно введення нових функцій управління і оркестрації MANO (Management and Orchestration). Необхідно також додавання рівня віртуалізації, за допомогою якого програмний рівень, який реалізує функціонал мережевого елемента, відділяється від апаратного рівня (процесора, пам'яті, мережевих інтерфейсів).

При управлінні віртуалізуванням мережних функцій VNFs необхідно враховувати наступні фактори:

- мультивендорної реалізація цих віртуалізованих функцій;
- управління життєвим циклом і взаємодія цих функцій;
- управління розподілом апаратних ресурсів;
- контроль використання;
- налаштування VNFs;
- взаємозв'язок віртуальних функцій для реалізації сервісу;

- взаємодія з OSS (Operation Support System) і білінгом.

Сукупність компонентів NFV вимагає стандартизації інтерфейсів взаємодії між ними, координації ресурсів, запровадження відповідного рівня абстракції для їх специфікації, тобто того, що робить еталонна модель. Ця еталонна модель повинна гарантувати, що розгорнуті таким чином VNFs не прив'язані до конкретного устаткування і не вимагають адаптації до конкретного середовища того чи іншого вендора. Вона повинна надавати дійсно еталонну архітектуру, якій можуть слідувати будь-які вендори, оператори, сервіс-провайдери для забезпечення узгодженості та однаковості в методологіях розгортання своїх рішень.

Така еталонна модель NFV MANO створена в ETSI NFV ISG і встановлює стандарти управління і оркестрації всіх ресурсів NFV, включаючи обчислювальні, мережеві ресурси, системи зберігання даних, віртуальні машини та ін. Мета NFV MANO - створення інфраструктури для гнучкого розгортання та управління мережевими функціями, як існуючих мережевих елементів, так і нових віртуальних мережевих пристроїв.

Еталонна модель ETSI (European Telecommunications Standards Institute) NFV MANO представлена на рисунку 2.2.1.1.

Необхідно відзначити важливу властивість даної моделі, яка пов'язана з управлінням NFV. Традиційно для управління мережевими елементами потрібна система EMS/NMS (Element Management System/Network Management System). Більш високим рівнем управління мультивендорної інфокомунікаційної мережею є OSS (Operation Support System), яка об'єднує EMS/NMS різних вендорів і виконує ряд інших функцій. Обидві вони присутні в моделі NFV (на рисунку 2.2.1.1 системи EMS/NMS показані в складі OSS), а крім них є ще три керуючих елемента, специфічних виключно для NFV. Це VIM (Vi Improved) Manager, VNF Manager і Orchestrator. Всього виходить п'ять елементів управління. [2]

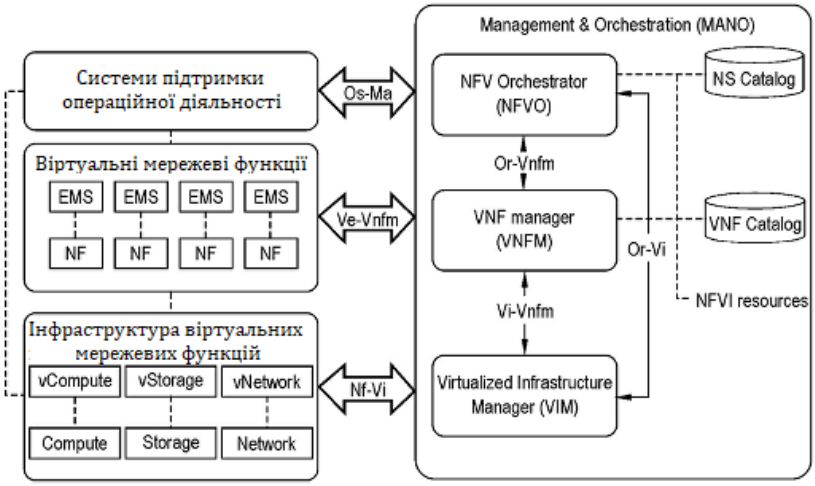

Рисунок 2.2.1.1 Еталонна модель ETSI NFV MANO

У правій частині рисунку 2.2.1.1 показані чотири компоненти логічної структури NFV MANO - три менеджера - Virtualized Infrastructure Manager (VIM), VNFManager (VNFM), NFV Orchestrator (NFVO) і набір репозиторіїв NS Catalog, VNF Catalog і NFVI Resources.

2.2.2 Елементи еталонної моделі NFV

Еталонна модель, яку представлено на рисунку 2.2.1.1 дозволяє уточняти логічну архітектуру NFV і її складові частини - NFVI, VFN, MANO.

1) Інфраструктура віртуалізації мережних функцій NFVI (NFV Infrastructure) є основою загальної архітектури NFV і включає апаратні засоби NFV і віртуальні засоби (рисунок 2.2.1.1).

До апаратних засобів NFV відноситься обладнання загального призначення: сервери, сховища пам'яті і мережеві пристрої. Це фізична частина інфраструктури NFVI, на якій виконуються віртуальні мережеві функції VNF. Сховища можуть бути локальними або розподіленими мережевими NAS (network-attached storage). Мережеве обладнання включає пули мережевих інтерфейсних плат і портів, які можуть використовуватися VNFs. Жоден з цих апаратних засобів NFV не створюється спеціально для якихось певних мережевих завдань, всі елементи є стандартними апаратними пристроями

загального призначення. Більш того, ці апаратні функції можуть масштабуватись на декількох взаємопов'язаних пристроях, необмежених одним фізичним хостом, загальним місцем розташування або точкою присутності POP (point of presence).

При цьому мережеве обладнання, пов'язане з фізичним місцем розташування і з'єднує між собою пристрої зберігання, обчислювальні пристрої та інші мережеві елементи: оптичні транспондери, бездротовим обладнанням, комутатори і тому подібне - також вважається частиною NFVI. Але ці службові мережеві пристрої не є частиною пулу, який виділяється для VNF.

До складу NFVI крім апаратних засобів включається віртуальний шар, який складають операційні системи і спеціальні програмні платформи - гіпервізор (hypervisor), що працюють на фізичних апаратних засобах і утворюють разом з ними віртуальні машини, віртуальне сховище та послуги віртуальної мережі для верхнього рівня VNF. Фактично гіпервізор виконує відділення програмних засобів від апаратних, тобто дає можливість запускати програми незалежно від обраного обладнання. Наприклад, операційна система може працювати на будь-якому фізичному сервері, який призначається для цієї мети в даний момент.

2) Віртуалізовані функції VNFs (Virtualised Network Functions) реалізуються за допомогою мережевого програмного забезпечення поверх NFVI. Власне, саме на рівні VNF і здійснюється віртуалізація мережевих функцій.

Згідно з концепцією NFV віртуалізована реалізація мережевих функцій створюється таким чином, щоб вона могла працювати на будь-якому обладнанні, що має достатні обчислювальні ресурси, пам'ять і мережеві інтерфейси. Однак сама VNF не знатиме, що обладнання, на якому вона працює, насправді є віртуальною машиною.

Для реалізації мережевої служби VNF можуть бути розгорнуті як автономні віртуальні функції, так і комбінації декількох VNFs.

В останньому випадку протоколи взаємодії між різними віртуальними мережевими елементами всередині VNF не повинні нічого "знати" про способи реалізації інших функцій: наприклад, VNF, що реалізують функції firewall, маршрутизації, NAT, Mobility Management Entity (MME), Evolved Packet Core (EPC), Serving Gateway (SGW) та інші обмінюються даними один з одним, не знаючи про те, віртуальні вони чи працюють на виділених фізичних пристроях. [2]

3) Система управління та оркестровки MANO (NFV Management and Orchestration) взаємодіє як з блоками NFVI, так і з VNF і відповідає за комплексне управління і моніторинг. На рівень MANO передається керування всіма ресурсами на рівні інфраструктури. Крім того, там же ресурси розподіляються між VNF, створюються і видаляються.

4) Оркестратор NFVO (Network Functions Virtualization Orchestrator) генерує, обслуговує, а потім припиняє роботу мережевих сервісів (функцій) VNF, а також ініціює створення комплексної служби з кількох VNF.

Оркестратор NFVO відповідає і за адміністрування глобальних ресурсів NFVI. При цьому NFVO не взаємодіє безпосередньо з VNF, а тільки через VNFM і VIM. З цієї точки зору роль NFVO не настільки очевидна і може здатися додатковим буфером між VIM і VFNM. Однак NFVO грає важливу роль при розгортанні комплексних служб. Прикладами подібних комплексних служб можуть бути віртуальна базова станція або віртуальний домен опорної мережі vEPC.

У мережі ці елементи можуть бути представлені обладнанням або від одного, або від різних вендорів, на базі яких і потрібно створити комплексну (end to end) службу із задіянням кількох VNF.

Для цього якраз і потрібен оркестратор послуг NFVO, що забезпечує комунікацію з усіма VNF, що беруть участь у створенні комплексної служби.

5) Менеджери VNFM в MANO займаються управлінням VNFs. Кожен такий менеджер управляє роботою однієї або декількох VNF. Наприклад, він управляє життєвим циклом функцій VNF, тобто запускає, обслуговує і зупиняє

роботу VNF. Таким чином, VNFM виконує схожі з EMS функції, але тільки через референсну точку VeNf-Vnfm, показану в еталонної моделі ETSI NFV на рисунку 2.2.1.1.

6) Менеджер віртуалізованої інфраструктури VIM. Для управління NFVI до складу MANO включений функціональний блок управління, званий Virtualized Infrastructure Manager (VIM). Йому делегується відповідальність за управління всім NFVI, тобто обчислювальним обладнанням, сховищем і мережевим обладнанням, а також програмним забезпеченням NFVI, які реалізують рівень віртуалізації.

Оскільки VIM безпосередньо управляє апаратними ресурсами, він вирішує повний набір завдань технічного обліку (Inventory), тобто виконує повну інвентаризацію цих ресурсів і їх експлуатаційних атрибутів (таких як живлення, стан працездатності, доступність, статистика використання). VIM також керує рівнем віртуалізації і контролює, як рівень віртуалізації використовує обладнання), управляє інформацією репозиторіїв про апаратні і програмні ресурси NFVI, груповими політиками безпеки, виконує збір інформації про продуктивність і відмовах устаткування, програмного забезпечення і віртуальних ресурсів, передає результати вимірювання продуктивності та інформації про події і відмови, що відноситься до віртуальних ресурсів.

7) Репозиторії даних MANO. Архітектура ETSI визначає також репозиторії в складі MANO, наведені в правій частині рисунку 2.2.1.1. Ці репозиторії зберігають необхідну інформацію для організації VNFs, а також відомості про доступні ресурси, які можуть бути задіяні для цього.

NS каталог (Network Service Catalog) використовується функціональним блоком NFVO і являє собою набір репозиторіїв, що визначають параметри для наскрізного розгортання мережевого сервісу - деякого об'єднання мережевих функцій, пов'язаних між собою з метою надання кінцевим користувачам цього мережевого сервісу. Таким чином, опис мережевого сервісу є описом складових

його зв’язку із мережею, їх зв'язності і топології, а також специфікаціями для їх розгортання.

8) Каталог VNF є сховищем VNF-пакетів. Кожен пакет VNF відповідає дескриптору VNFD (VNF Descriptor), який використовується для визначення параметрів розгортання VNF, таких як вимоги до ресурсів процесора, пам'яті і сховища, політики, події життєвого циклу. Це можуть бути підвищені вимоги до продуктивності процесора і вельми скромні до пам'яті, якщо мова йде про маршрутизатор, наприклад. Інформація з каталогу VNF затребувана як NFVO, так і VNFM. При цьому NFVO використовує його для управління життєвим циклом VNF. Зокрема, для створення нової VNF, яка є частиною мережевого сервісу, передається вимога на створення з відповідною інформацією з каталогу NS (наприклад, VLD) в функціональний блок VIM. Останній на підставі цієї інформації вибирає і виділяє необхідні ресурси. [2]

2.3 Хмарно реалізована мультимедійна IP підсистема на основі архітектури віртуалізації системних функцій (NFV)

Віртуалізація мережевих елементів та їх реалізація як програмних компонентів є лише аспектом процесу клаудифікації. Після того, як компоненти віртуалізуються, вони можуть розміщуватися поверх віртуальних машин, і всі хмарні принципи можуть бути застосовані і до них, наприклад, поняття еластичності. Еластична інфраструктура здатна автоматично збільшувати або зменшувати кількість використовуваних ресурсів без обмежень. Для автоматичного включення еластичності мережевих елементів необхідно визначити та виконати певну процедуру масштабування.

Залежно від програмного забезпечення екземпляра IMS, різні функції можуть бути розділені та згруповані, як того вимагає конкретний екземпляр. Існує три варіанти розгортання IMS у хмарному середовищі:

1) Віртуалізована IMS (Virtualized-IMS (vIMS));

Virtualized-IMS (vIMS) - тип архітектури, в якому кожен функціональний об'єкт IMS 3GPP повністю співпадає з функціональними блоками віртуальної мережі;

Тобто дана архітектура вимагає реалізації кожного функціонального об'єкта IMS 3GPP на одній віртуальній машині. Зокрема, це основний архітектурний підхід, який виконується під час віртуалізації мережевих компонентів. У цьому підході інтерфейси із зовнішніми компонентами не змінюються і є стандартизованими 3GPP. Еластичність реалізується за допомогою процедур, уже стандартизованих 3GPP.

2) Розділена IMS (Split-IMS);

Split-IMS - тип архітектури, в якому кожен компонент IMS розбивається на кілька підкомпонентів для того, щоб бути розгорнутим поверх декількох хостів і контейнерів;

На рисунку 2.3.1 показана можлива архітектура віртуалізації та розподілу функцій мережі. Балансер мережевих функцій (NFBalancer) є точкою входу для вхідних запитів. NFBalancer поширює запити на кілька NFWorkers. Стан абонента підтримується у зовнішньому функціональному елементі під назвою SharedMemory.

Інтерфейс між користувачем та NFBalancer стандартизований, тоді як інші інтерфейси є специфічними для кожної практичної реалізації і не впливають на взаємодію. Між NFBalancer та NFWorker та між NFWorker та SharedMemory можна використовувати будь-який інтерфейс. Розділивши три рівні, передбачені наступні переваги: зменшення складності балансирів навантаження; переміщення стану у зовнішню спільну пам'ять забезпечує високу еластичність виконання функцій процедури; ізоляція елементів NF в приватний, виділений пул. [17]

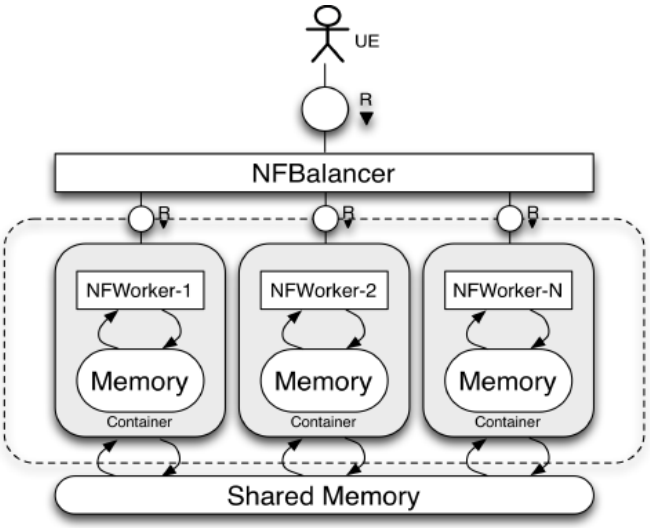

Рисунок 2.3.1 Концепт архітектури розділеної IMS (Split-IMS)

Цей тип конфігурації може використовуватися для розгортання одного мережевого елементі в одному датацентрі.

3) З’єднувальна IMS (Merge-IMS)

Merge-IMS - тип архітектури, в якому компоненти об’єднуються в менші компоненти, що забезпечує малу затримку і функціонально зменшену обробку для зовнішніх запитів одним і тим же функціональним блоком віртуальної мережі.

Остання архітектурна модель описує можливість об'єднання різних функціональних об'єктів IMS з метою зменшення складності, що робить її дуже простою архітектурою. Зокрема, ця віртуальна машина називається IMS VM, як зображено на рисунку 2.3.2, і містить чотири основні функції: P-CSCF, SCSCF, I-CSCF та HSS.

На рисунку 2.3.2 показана повна архітектура сценарію з’єднувальної-IMS.

Дана архітектура має три основні суб'єкти:

• IMSLocator: це об'єкт, необхідний для присвоєння абонентам певного екземпляру IMS-VM під час процедури реєстрації та локалізації екземпляру IMS-VM під час фази виявлення. Цей об'єкт повинен розкрити інтерфейси Gw і Mw, стандартизовані 3GPP;

• Пул IMS-VM: це пул елементів IMS-VM. Він керується сервісним оркестратором, який вирішує, чи залучати нові IMS-VM на основі деяких умов, визначених EEU;

• SharedDB: спільна база даних між різними екземплярами IMS VM. Він необхідний для зберігання інформації про абонентів.

З точки зору реалізації мультимедійної IP підсистеми у хмарному середовищі використання архітектури розділеної IMS дозволить забезпечити масштабованість, гнучкість та оптимальне використання ресурсів мережі. Дана архітектура заснована на використанні багатоетапної подійно-орієнтованої архітектури SEDA (Staged-event driven architecture), що дозволяє розгортати у хмарному середовищі інфокомунікаційні мережі з підтримкою роботи великої кількості користувачів. [17]

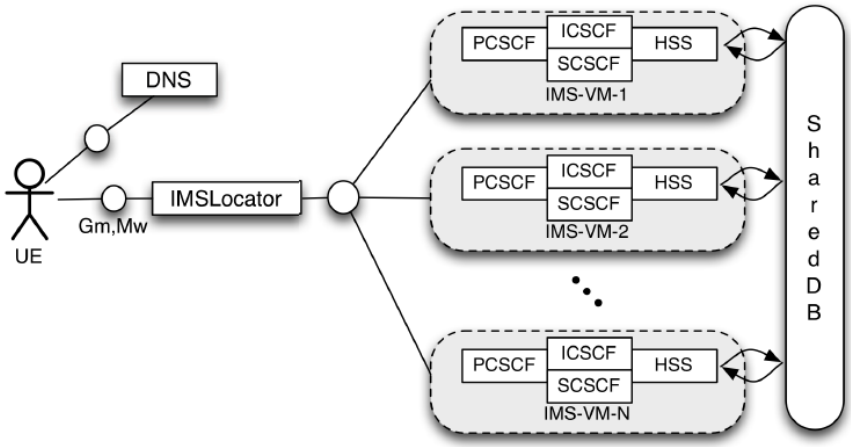

Рисунок 2.3.2 Архітектура з’єднувальної-IMS (Merge-IMS)

2.4 Висновки до розділу 2

У розділі №2 досліджуються способи віртуалізації концепції IMS у хмарному середовищі. Проводячи аналіз наведеної інформації та поставленої мети роботи встановлено, що в рамках магістерської дисертації оптимальним рішенням розгортання IMS є модель платформи як послуги (PaaS).

На основі даної моделі обрано і спосіб реалізації IMS у «хмарі» – розділена IMS, або ж split-IMS. Даний спосіб створює можливість до розгортання функціонального елементу мережі розбиваючи його на декілька

компонентів і засновується він на принципі багатоетапної подійно-орієнтованої архітектури SEDA. При реалізації SEDA додатки складаються з системи подійно-орієнтованих етапів, з'єднаних певними чергами, кожен етап реалізує в собі певний програмний компонент, що в процесі роботи мережі буде постійно виконувати обробку запитів, що до нього надходять.

Збираючи інформацію про кожен компонент мережі за допомогою своєї системи моніторингу і користуючись зворотнім зв’язком SEDA слідкує за роботою кожного компоненту системи і проводить автоматичний динамічний контроль коригуючи їхню роботу, тобто виділяючи стільки ресурсів на послугу, скільки необхідно для її постійної оптимальної роботи.

РОЗДІЛ 3. АДАПТИВНА МОДЕЛЬ БАГАТОЕТАПНОЇ ПОДІЙНО-ОРІЄНТОВАНОЇ АРХІТЕКТУРИ SEDA

В даному розділі розглядається багатоетапна подійно-орієнтована архітектура (SEDA), яка покликана підтримувати високі вимоги щодо паралелізму та спрощувати побудову послуг шляхом гнучкого моніторингу та як наслідок забезпечувати гнучкий інжиніринг навантаження на елементи мережі. У SEDA додатки складаються з системи подійно-орієнтованих етапів, з'єднаних певними чергами. Ця архітектура дозволяє послугам бути готовими до високих навантажень, запобігаючи перевантаженню наявних ресурсів, коли попит перевищує ємність послуги. Так SEDA використовує набір динамічних контролерів ресурсів, щоб підтримувати етапи в робочому режимі, незважаючи на великі коливання навантаження.

3.1 Концепція багатоетапної подійно-орієнтованої архітектури SEDA

Розглядаючи концепцію SEDA необхідно виділити ряд понять, які є основоположними в даній архітектурі.

Етап, або ж стадія, є фундаментальною одиницею обробки в рамках концепції SEDA. Це самостійний компонент програми, що складається з обробника подій, черги вхідних подій та множини потоків. Кожен етап керується одним або декількома контролерами, які впливають на споживання ресурсів, планування, розподіл потоків та контроль надходження заявок.

Роль потоків в стадіях полягає у витягуванні подій із вхідної черги та викликанні відповідного їм обробника. Обробник подій, в свою чергу, обробляє кожну партію подій, які на нього надходять, та відправляє нуль або більше надлишкових подій у черги подій інших етапів. Кожна подія, що вже оброблена етапом, представляє собою структуру даних, що відповідає одному запиту клієнта мережі до інтернет-сервісу, наприклад, HTTP-запит на веб-сторінці. Однак події можуть представляти собою й іншу інформацію, що стосується

роботи сервісу, наприклад, у випадку роботи клієнта з інтернет-сервісом, встановлення мережевого з'єднання з інтернет ресурсом. Так модель програмування SEDA працює з партіями подій, а не окремо з кожною з них. Це дозволяє додатку використовувати всі доступні йому ресурси для обробки декількох подій разом, що може призвести до підвищення продуктивності. [18]

Обробники подій, у свою чергу, реалізуються розробником додатку для кожного етапу, який представляє основну логіку обробки послуги для цього етапу. Обробник подій - це звичайна функція, яка приймає партію подій як вхідну, оброблює ці події та (необов'язково) переносить вихідні події на інші етапи. Сам обробник подій не має прямого контролю над потоками в середині етапу, чергою введення та іншими аспектами управління ресурсами та планування. У деякому сенсі обробники подій - це пасивні компоненти, які викликаються системою виконання у відповідь на прибуття події, а час виконання визначає, коли і як необхідно викликати обробник подій.

Основний мотив для такої організації роботи даного компоненту - це відокремлення проблем між собою: відокремлюючи логіку основної програми від управління потоками та планування, система управління може контролювати роботу обробника подій, що необхідно для реалізації різних зовнішніх політик управління ресурсами. Наприклад, кількість та впорядкованість подій, переданих обробнику подій, контролюється під час виконання, як і пріоритет розподілу та планування потоків, що діють на етапі. Мета архітектури SEDA полягає у спрощенні дизайну сервісу, забезпечивши управління ресурсами в загальних рамках, що звільняє розробників послуг від значної складності управління завантаженням. Однак такий підхід не позбавляє додатків усякого контролю за функціонуванням служби: на етапі програми можуть приймати рішення про те, щоб вимагати визначення пріоритетності чи якості обслуговування. [18]

1) Модель потоків

Потоки є основним механізмом паралелізму в SEDA, але їх використання обмежується їх невеликою кількістю на етапі, порівняно з виділенням окремого потоку на кожен запит у системі. Даний підхід надає ряд переваг.

По-перше, використання потоків послаблює обмеження на те що весь код обробки подій не повинен бути блокуючим, що надає права обробнику блокувати події, або бути вилученим з системи при необхідності. У традиційній подійно-орієнтованій системі, всі операції блокування повинні фіксувати стан процесу обробки та відновлювати його після завершення операції. У SEDA потоки діють як неявні продовження, автоматично фіксуючи стан виконання через операції блокування. Так додатки SEDA створюють явні продовження, коли вони відправляють подію на інший етап. Перевага цього підходу полягає в тому, що управління продовженням може виконуватись додатком тоді, коли це найзручніше (тобто на перетинах границь етапу, при переході із одного в інший), а не в будь-який час під час роботи обробника події. Створення таких продовжень, як правило, складається з виділення невеликої за розміром структури даних про подію, що необхідно для надання інформації про дану операцію, яка буде виконуватись наступним етапом, і встановити стан цього етапу. У багатьох випадках одна і та ж структура подій може передаватися разом від етапу до етапу, оскільки етапи виконують перетворення свого стану відповідно до події.

Блокування, в свою чергу, ефективно знижує кількість активних потоків на етапі, тим самим стабілізуючи швидкість обробки запиту на етапі. Для підтримки високої паралельності додаткові потоки можуть бути виділені на етап, який знаходиться в режимі блокування. Однак, щоб уникнути зайвого використання потоків (відповідно зменшення продуктивності), необхідно, щоб кількість потоків, які необхідні для перекриття операцій блокування, була не надто великою. Це означає, що більшість операцій блокування повинні бути короткими, або що операції тривалого блокування повинні бути рідкісними.

По-друге перевага цієї моделі потоків полягає в тому, що SEDA може розраховувати на операційну систему для чіткого планування етапів на апаратному забезпеченні, на відміну від монолітного дизайну керованого подіями, в якому адміністратор послуг повинен визначити порядок і політику диспетчеризації подій. У SEDA використання потоків дозволяє виконувати етапи послідовно або паралельно, залежно від операційної системи, системи потоків та планувальника завдань. Зважаючи на малу кількість потоків, використовуваних у типовому додатку SEDA, прийнятним є використання потоків рівня ядра, які (на відміну від багатьох партій потоків користувачів) можуть використовувати декілька процесорів у багатопроцесорній системі.

Для SEDA альтернативою до керування операційною системою було б планування обробників подій безпосередньо на фізичних процесорах в системі. Складність при такому підході полягає в тому, що це змушує SEDA дублювати більшість функціональних можливостей планувальника ОС та обробляти всі аспекти блокування, попередження, встановлення пріоритетів тощо. Однак такий підхід забезпечить великий ступінь контролю над впорядкуванням і пріоритетністю окремих запитів або етапів у системі. Обробники подій можуть перенести події на інший етап, спочатку отримати шлях до черги вхідних подій цього етапу (через процедуру пошуку системи), а потім викликати операцію запитання на цій черзі. [18]

2) Додатки як мережі етапів

Додаток SEDA будується як мережа стадій, з'єднаних чергою подій. Обробники подій можуть переносити події в черги інших етапів, спочатку отримуючи дескриптор вхідної черги подій в цю стадію (через передбачену системою процедуру пошуку), а потім викликаючи операцію зміни черги.

Важливим аспектом черг в SEDA є те, що вони є об'єктом контролю допуску. Тобто, операція зміни черги може бути відхилена у зв’язку з реалізацією певної політики управління ресурсами, наприклад, понаднормове перевищення часу реагування. Відхилена операція негайно подається на початкову стадію, наприклад, сигналізуючи про це кодом помилки. Відхилення

запиту виконує функцію сигналу про перевантаження додатків, і його слід використовувати для адаптації роботи системи. Слід відзначити, що механізм контролю надходження та відповідь на відмову значною мірою залежить від політики управління перевантаженням та самої програми.

Мережа етапів може бути побудована або статично (де всі етапи та з'єднання між ними відомі під час компіляції чи завантаження), або динамічно (дозволяючи етапи додавати та видаляти під час виконання). У кожного підходу є переваги та недоліки. Статична побудова мережі дозволяє адміністратору послуг (або автоматизованому інструменту) міркувати про правильність структури графіків; наприклад, чи фактично обробляються типи подій, що генеруються одним етапом, етапами нижче від нього. Статична побудова може також дозволити оптимізацію часу компіляції, наприклад, може пришвидшити знаходження «петлю» між двома етапами, ефективно поєднуючи два етапи в один і дозволяючи використовувати код з одного етапу на інший.

Динамічна побудова мережі забезпечує набагато більшу гнучкість у розробці додатків, вказуючи нові етапи, які потрібно додавати до системи за необхідності. Наприклад, якщо якась функція послуги рідко використовується, відповідні етапи можуть бути ініційовані лише за запитом.

3) Проблеми структуризації додатків

Будь-який додаток на базі SEDA, базується на ряді факторів. Важливі проблеми, які слід враховувати, включають продуктивність, контроль навантаження та модульність коду.

Принциповим при побудові додатків є прийняття рішенням про те, чи повинні два модулі коду обробки запитів зв’язуватись через чергу (тобто реалізовуватись як окремі етапи) або безпосередньо через виклик функції (реалізовуватись як один етап).

З причин продуктивності, бажано використовувати прямий виклик функції, щоб уникнути проблем синхронізації та планування витрат ресурсів на відправку події в чергу. Однак використання черг має ряд переваг, включаючи ізоляцію, незалежне управління навантаженням та модульність коду. Введення

черги між двома кодовими модулями розділяє їх виконання, забезпечуючи чітку межу розподілення управління. Виконання потоку обмежується на поточному етапі, що обмежує час його виконання і використання ресурсів, що витрачаються в межах свого власного етапу. Потік може передавати дані лише між етапами в якому він був спочатку та в який переходить подія. Як результат, споживання ресурсів кожним етапом можна контролювати незалежно один від одного, наприклад, здійснюючи контроль за надходженням до черги подій на етапі.

Черги також є прекрасним механізмом структурування складних додатків. Етапи приймають події певних типів та передають події певних типів, не потрібно зрівнювати між собою типи подій на прийомі та при передачі. Етапи складаються з використанням певного протоколу, а не просто з відповідності типу аргументів функцій та значень, таким чином допускаючи гнучкий діапазон композиції політик. Наприклад, етап може агрегувати дані протягом декількох подій з плином часу, лише іноді генеруючи "підсумкові події" недавньої активності. Узгодження типів подій через інтерфейс черги не повинно бути жорстким, наприклад, етап може прозоро проходити вздовж типів подій, які він не розуміє.

Черги також полегшують налагодження та аналіз продуктивності послуг, що традиційно викликає труднощі в складних багатопотокових серверах. Код моніторингу може бути доданий до вхідних та вихідних точок кожного етапу, що дозволяє адміністратору системи профілювати потік подій через систему. [18]

4) Динамічне управління ресурсами

Ключова мета, що сприяє простоті сервісної інженерії, - захистити програмістів від складності налаштування продуктивності. Щоб утримати кожен етап у своєму ідеальному режимі роботи, SEDA використовує динамічне управління ресурсами, автоматично адаптуючи поведінку кожного етапу на основі спостережуваних показників ефективності та рівня використання. Абстрактно, контролер дотримується характеристик часу виконання етапу та

коригує параметри розподілу та планування для досягнення цільових показників продуктивності. Контролери можуть працювати або з повністю локальними знаннями про певний етап, або на основі глобального стану.

У рамках SEDA можливий широкий спектр механізмів управління ресурсами. Одним із прикладів є налаштування кількості потоків, що виконуються на кожному етапі. Якщо всі операції на етапі є блокуючими, то для етапу потрібно не більше одного потоку на центральний процесор. Однак, враховуючи можливість коротких операцій блокування, для підтримання паралелізму можуть знадобитися додаткові потоки. Аналогічно, виділення додаткових потоків на етап приводить до того, що цей етап надає більш високий пріоритет, ніж інші етапи, в тому сенсі, що він має більше можливостей для виконання. У той же час важливо уникати перерозподілу потоків, так як це призводить до погіршення продуктивності. Тому ідеальна кількість потоків на етап базується на вимогах паралелізму етапу та загальній поведінці системи. Замість того, щоб виділити статичну кількість потоків для кожного етапу, SEDA використовує динамічний контролер пулу потоків для автоматичної настройки розподілу потоків.

Динамічне управління в SEDA дозволяє програмі адаптуватися до мінливих умов, незважаючи на конкретні алгоритми, що використовуються в основі операційної системи. У деякому сенсі контролери SEDA наївно ставляться до політики управління ресурсами ОС. Наприклад, контролер розміщення пулу потоків SEDA не обізнаний з політикою планування потоків ОС; скоріше, це впливає на розподіл потоків на основі зовнішніх спостережень за роботою програми. Хоча в деяких випадках бажано посилити контроль над базовою ОС - наприклад, забезпечити гарантії якості обслуговування для окремих етапів або потоків. [18]

5) Захист від перевантаження

Крім налаштування параметрів виконання, ще одна форма управління ресурсами в SEDA - це контроль над перевантаженням. Тут мета полягає в тому, щоб не допустити показника значно зниженої продуктивності під

великим навантаженням через надмірне використання ресурсів. По мірі того, як сервіс наближається до насичення, час відповіді, виставлений на запити, може зростати експоненціально.

Так захист від перевантаження в SEDA здійснюється завдяки використанню розподіленого контролю доступу на кожному етапі, який може бути використаний для реалізації широкого спектру політик. Як правило, застосовуючи контроль над доступом, система може обмежувати швидкість, з якою цей етап приймає нові запити, дозволяючи ізолювати вузькі місця і розподілити навантаження, яке на них припадає. Проста політика контролю за надходженням може полягати в застосуванні фіксованого порогу до черги подій кожного етапу. Визначити ідеальні пороги для досягнення найвищої ефективності роботи системи повинен сам провайдер послуг.

Найкращий підхід полягає в тому, щоб поетапно контролювати їхню ефективність (наприклад, розподіл часу та відповіді) та викликати відмову в обслуговуванні вхідних подій, коли певний поріг продуктивності перевищено. Крім того, контролер доступу може призначити важливість кожної події в системі та визначити пріоритетні події низької важливості (наприклад, недорогі статичні запити веб-сторінок) над подіями високої важливості (наприклад, важкі динамічні сторінки). SEDA дозволяє налаштувати політику контролю доступу для кожного окремого етапу, а контроль доступу можна відключити на будь-якій стадії.

Основною властивістю дизайну послуг SEDA є те, що етапи повинні бути підготовлені для вирішення питання про відмову операції переводу події з черги в чергу. Відхилення даної операції вказує на те, що відповідний етап перевантажений і сервіс повинен використовувати цю інформацію для адаптації. Ця явна вказівка на перевантаження відрізняється від традиційних службових конструкцій, які розглядають перевантаження як винятковий випадок, для якого додатки мають мало вказівок або контролю. У SEDA управління перевантаженням є першокласним примітивом у програмній моделі.

Так відмова в обслуговуванні події з черги не означає, що запит користувача відхилено системою. Швидше за все, відповідальний за це етап, який отримує відмову про додавання події від черги, виконує якусь альтернативну дію. Ця дія залежить від логіки обслуговування. Наприклад, якщо статичний запит веб-сторінки відхилено, зазвичай достатньо надіслати клієнту повідомлення про помилку із зазначенням, що послуга перевантажена. Однак якщо запит стосується складної операції, такої як здійснення біржової торгівлі, доведеться надати відповідь іншими способами, наприклад, попросити користувача здійснити запит повторно, або ж провести його пізніше. [18]

Підсумовуючи вище наведену інформацію можна відзначити, що архітектура багатоетапної подійно-орієнтованої архітектури SEDA має наступні властивості:

- Ефективний, подійно-орієнтований паралелізм:

Для підтримки масивних ступенів паралелізму, SEDA покладається на подійно-орієнтовані методи для передачі кількох потоків через систему. Таким чином використовується невелика кількість потоків, замість використання одного потоку на один запит.

- Динамічне об'єднання потоків:

Щоб послабити вимоги до жорсткого планування, щодо одночасності ведення подій, SEDA використовує набір об’єднаних потоків, один на етап, для можливості управління над їх виконанням.

- Структуровані черги для модуляції коду та управління навантаженнями:

Розбивши додаток на набір етапів із чіткими чергами між ними, дизайнери додатків можуть зосередитись на логіці обслуговування та управлінні паралельністю на окремих етапах, згодом «підключивши» їх до повної послуги. Черги забезпечують точку контролю над потоком запитів у послузі, оскільки запити, що проходять через етапи, можуть перевірятися та керуватися програмою.

- Управління ресурсами самоналаштування:

Замість того, щоб надати апріорне знання вимог до ресурсів програми та характеристик завантаження клієнта, SEDA може використовувати зворотний зв'язок та управління для автоматичної настройки різних параметрів використання ресурсів у системі. Наприклад, система визначає кількість потоків, виділених для кожного етапу, виходячи із сприйнятих вимог одночасності, а не спираючись на жорстко закодоване значення, встановлене програмістом або адміністратором. [18]

3.2 Базові рішення для побудови моделі обробки заявок

В даному пункті дипломної роботи наводиться набір шаблонів дизайну, які можуть бути застосовані при побудові систем на базі SEDA. Такі шаблони - це специфікації, що використовуються для рішення стандартних проблем, що виникають в процесі розробки системи на базі SEDA, а також для досягнення цією системою певних показників продуктивності та надійності.

1) Шаблон обгортання

Шаблон обгортання розміщує чергу перед набором потоків, що виконують обробку завдань, тим самим "обгортаючи" логіку обробки завдання на етап SEDA. Кожен потік обробляє одне завдання через деяку кількість кроків і в той же час може бути заблокованим. Слід зазначити, що в подальшому описі пункту "завдання" визначається як основний блок роботи, яка повинна виконуватись у відповідь на запит клієнта. Абстрактно завдання складається з набору етапів обробки, які можуть включати обчислення, введення / виведення, мережевий зв’язок тощо.

Операція обгортання робить отриманий етап надійним для подальшої роботи, оскільки кількість потоків всередині етапу тепер можна зафіксувати за значенням, що запобігає погіршенню продуктивності потоку, а додаткові завдання, які не можуть обслуговуватись цими потоками, накопичуватимуться в черзі.

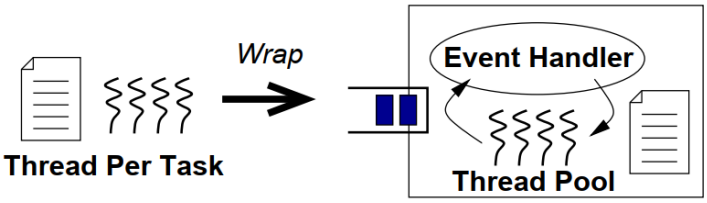

Рисунок 3.2.1 Шаблон обгортання (The Wrap design pattern)

Обертання всієї логіки обслуговування на одному етапі, як правило, є занадто грубим для забезпечення адекватних властивостей системи та управління навантаженням. Це пояснюється тим, що такий підхід надає контроль лише над двома змінними: кількості потоків на етапі та політиці контролю доступу до черги запитів. Для управління над вузькими місцями у послузі, тобто в тих місцях де ресурсів недостатньо для оптимальної роботи служби, необхідно контролювати потік запитів всередині сервісу, наприклад, ідентифікуючи ті запити, які спричиняють погіршення продуктивності, та виконувати розподіл навантаження що припадає на них. Однак, пул одного потоку та черга запитів не надають достатній «зовнішній» контроль над поведінкою логіки обслуговування. Декомпозиція послуги на кілька етапів дозволяє забезпечити незалежне розподілення ресурсів та управління навантаженням для кожного етапу. [18]

2) Тунелювання та розподіл

Шаблон тунелювання займає один етап і перетворює його в ланцюг з декількох послідовних етапів, вводячи між ними черги подій. Якщо обробка завдання на початковій стадії складається з етапу обробки, який має три кроки обробки S1, S2 та S3, тунелювання може використовуватись для створення трьох окремих етапів, кожен з яких виконує один крок. Дану концепцію показано на рисунку 3.2.2.

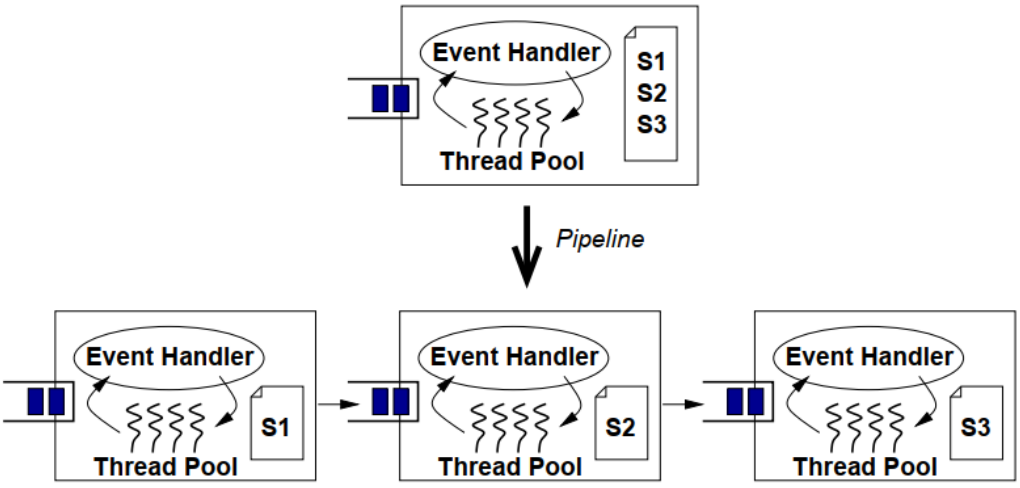

Рисунок 3.2.2 Шаблон тунелювання (The Pipeline design pattern)

Шаблон розподілу розділяє один етап і перетворює його в дерево декількох паралельних етапів. Ця трансформація, як правило, відбувається в точці розгалуження в процесі обробки завдання. Якщо етап виконує крок обробки S1 з подальшим відгалуженням або на S2, або на S3, тоді шаблон розподілу перетворює це у дерево трьох етапів, де S1 подає події на етапи S2 і S3. Дану концепцію показано на рисунку 3.2.3. [18]

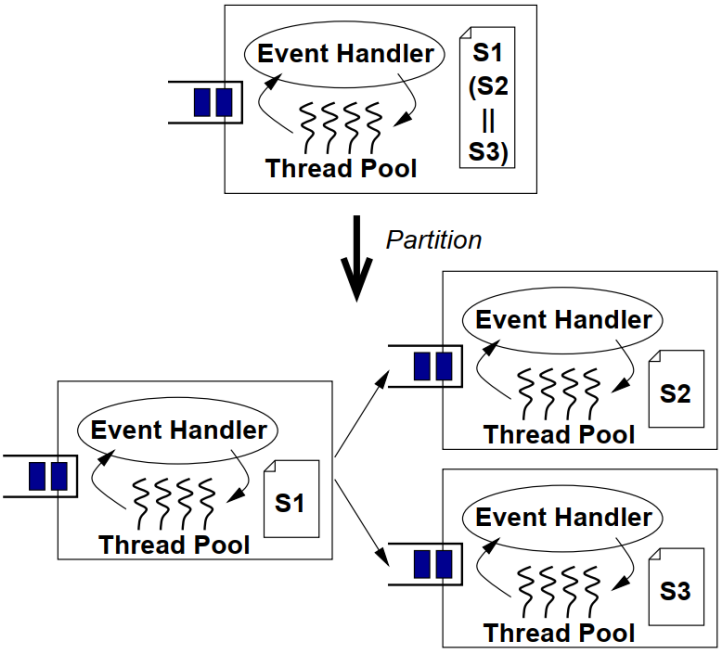

Рисунок 3.2.3 Шаблон розподілу (The Partition design pattern)

Шаблони тунелювання та розподілу мають ряд застосувань:

По-перше, вони можуть бути використані для виділення блокуючого або ресурсомісткого коду в його власну стадію, що дозволяє йому бути незалежним від інших етапів та самостійно управляти розподіленням навантаження. Розбиття обробки завдання на окремі етапи дозволяє пристосувати розмір пулу потоків до цього етапу і дозволяє дублювати цей етап через окремі фізичні ресурси для досягнення більшої паралельності. Цей підхід також дозволяє приймати рішення щодо управління навантаженням на основі потреб у кількості ресурсів кожного окремого етапу.

Логічною схемою дій шаблону розподілу при перетворенні одного етапу виконання логічних операцій на три виглядає наступним чином:

1. Виконайте завдання S1;
2. Якщо виконується деяка умова, виконайте завдання S2;
3. В іншому випадку виконайте завдання S3.

Наприклад, S2 виконує операцію блокування, яка потребує тривалого часу, і навантаження, яке створюється певним запитом, стає вузьким місцем, яке вимагає використання контролю доступу. Не застосовуючи шаблон розподілу, необхідно було б застосувати контроль доступу на одному етапі, виконуючи кроки S1; S2; S3, без будь-якої інформації про те, які запити реально виконувались на кроці S2. Однак, розміщуючи цей код на своєму окремому етапі, на запити, які проходять лише через етапи S1 і S3, це не впливає; контроль доступу може здійснюватися на S2 окремо від решти служби.

Застосування шаблонів тунелювання та розподілу може збільшити величину кешу даних та інструкцій, щоб уникнути зниження показника продуктивності. Кожен етап може обробляти партію завдань одразу, зберігаючи кеш інструкцій із власним кодом, а кеш даних - із будь-якими спільними даними, які використовуються для обробки партії завдань. Крім того, кожен етап має можливість обслуговувати вхідні завдання в порядку, який є оптимізованим для кешу. Наприклад, якщо черги обслуговуються в черговому

порядку, то завдання, що надійшли останніми, все ще можуть знаходитися в кеші даних. [18]

3) Шаблон комбінування

Шаблон комбінування поєднує два етапи в один етап. Це обернений шаблон до патернів тунелювання та розподілу, він використовується для агрегації обробки окремих ступенів. Одна з переваг цієї моделі - зменшення складності коду. Розглянемо набір з трьох послідовних етапів (S1, S2 та S3), що є результатом використання тунелювання, як показано на рисунку 3.2.4. Якщо S1 і S3 виконують тісно пов'язані дії, такі як відправлення та завершення асинхронної операції на S2 , то є сенс поєднувати логіку цих етапів в одну стадію. Наприклад, якщо S1 і S3 пов'язані з процесором, то обом етапам потрібно не більше одного потоку на процесор; немає необхідності у виділення окремого потоку для кожного етапу. Шаблон об’єднання має переваги і для локалості. Якщо S1 і S3 мають спільні структури даних або код, то поєднання їх обробки в один етап може покращити локальність кешу, що призведе до зростання продуктивності. [18]

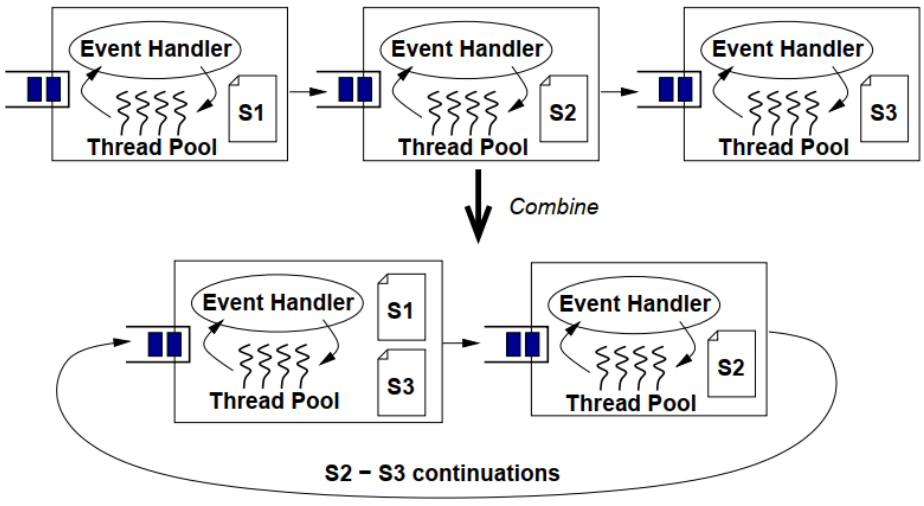

Рисунок 3.2.4 Шаблон комбінування (The Combine design pattern)

4) Шаблон копіювання

Шаблон копіювання реплікує заданий етап, вводячи межу відмови між двома копіями, можливо шляхом інстанціювання нового етапу на окремий

набір фізичних ресурсів. Тоді як тунелювання і розподіл використовуються для досягнення функціонального розкладання етапу, копіювання використовується для досягнення паралелізму та усунення несправностей. Канонічне використання даного шаблону полягає у розподілі етапів у додатку на основі SEDA через набір фізичних вузлів, наприклад, у кластері робочих станцій. Це дозволяє масштабувати обробку послуги, масштабуватися з кількістю вузлів кластеру, а також збільшити відмовостійкість, дозволяючи копіям етапу самостійно вимикатись на окремих вузлах.

Шляхом копіювання етапу через фізичні ресурси збільшується можливість комбінованої обробки реплік; це можна використати для усунення вузького місця в системі, виділивши більше ресурсів на копійований етап. Наприклад, якщо етап пов'язаний з процесором і не в змозі забезпечити очікувану продуктивність з наявними ресурсами, шаблон копіювання можна використовувати для використання додаткових процесорів у кількох вузлах кластера. Копіювання також може бути використане для введення межі відмови між копіями етапу, або запускаючи їх на окремих машинах, або навіть у різних адресних просторах на одній машині.

У етапній реплікації виникає складність у розподіленому управлінні стадіями. При несправності мережевого зв’язку в кластері може виникнути розділення, що викликає складнощі, якщо етапи, що знаходяться на різних вузлах кластера, потребують підтримання послідовного стану. Існує кілька способів уникнути цієї проблеми.

Наприклад, один з них - використовувати один з протоколів розподіленої узгодженості або групового членства. Інший - полягає у розробці з'єднання кластера для усунення розділення. Це підхід DDS та пошукової системи Inktomi. [18]

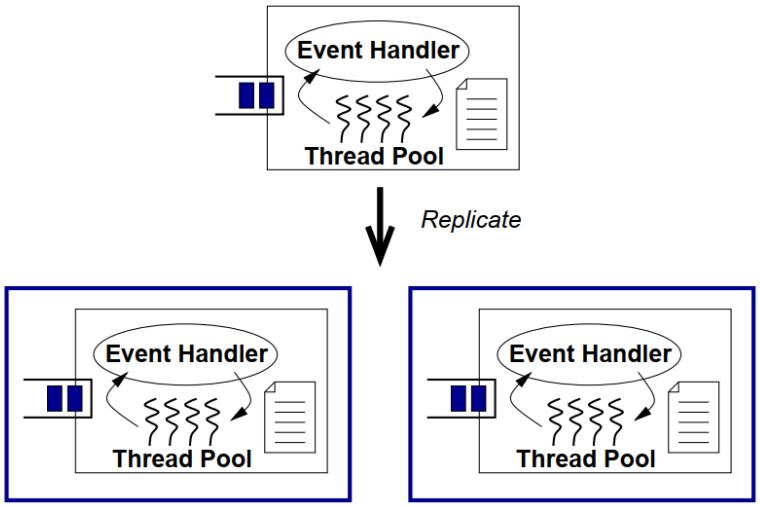

Рисунок 3.2.5 Шаблон копіювання (The Replicate design pattern)

3.3 Висновки до розділу 3

В розділі №3 розглядається багатоетапна подійно-орієнтована архітектура SEDA. В пунктах даного розділу наводиться інформація щодо організації та особливостей роботи даної концепції.

Аналізуючи наведену інформацію можна стверджувати, що на основі архітектури SEDA створюється програмне забезпечення, реалізація якого в високонавантаженій віртуалізованій інфокомунікаційній мережі дозволяє забезпечити оптимальну роботу кожного з елементів системи.

За допомогою шаблонів побудови моделі обробки заявок в даній концепції можна створити специфіковане програмне забезпечення, яке необхідне для вирішення завдань, що постають перед оператором інфокомунікаційних послуг в залежності від його потреб та вимог. Тобто архітектура SEDA дозволяє задовільнити широкий спектр вимог сервіс провайдерів, що в свою чергу забезпечує високу продуктивність, захист елементів мережі від перенавантаження та модульність програмного забезпечення.

В рамках даної роботи для проведення експериментального дослідження та, відповідно, створення програмного забезпечення використано шаблон

копіювання, що є досить зручним для перевірки продуктивності змодельованого елементу мережі та ресурсозатратності розгорнутої моделі.

РОЗДІЛ 4. МОДЕЛЮВАННЯ ФУНКЦІОНАЛЬНОГО ВУЗЛА МЕРЕЖІ IMS НА ОСНОВІ АРХІТЕКТУРИ SEDA

В даному розділі проводиться процес проведення експериментального дослідження на основі створеної моделі домашнього серверу абонентів HSS, на який надходить велика кількість вхідних заявок про надання послуг для користувачів. Програмна реалізація даного компоненту мережі IMS побудована на основі серверу RabbitMQ з використанням багатоетапної подійно-орієнтованої архітектури SEDA.

4.1 Функціональний вузол мережі – Home Subsriber Server

HSS являє собою централізоване сховище інформації про абонентів і послуги, і є еволюційним розвитком HLR (Home Location Register) з архітектури мереж GSM. У HSS зберігається вся інформація, яка може знадобитися при встановленні мультимедійного сеансу: інформація про місцезнаходження користувача, інформація для забезпечення захисту (аутентифікація і авторизація), інформація про профілі користувачів, про обслуговуючу користувача S-CSCF, про тригерні точки запитів для надання послуг. [4]

Місце HSS в структурі IMS показано на рисунку 4.1.1.

У HSS розміщується інформація про користувача, яка необхідна мережі для підтримки всіх функцій IMS, пов'язаних з обробкою виклику і встановленням сеансу. У числі іншого він містить інформацію про абонування послуг (часто звану профілями користувачів) і забезпечує дані, що вимагаються для аутентифікації і авторизації користувача доступу до послуг.

На основі записів в базі даних ідентифікується, до яких послуг користувач має доступ, з якою мережею він зараз з'єднаний. HSS підтримує також локалізацію користувача подібно домашньому регістру

місцезнаходження HLR і регістру місцезнаходження VLR в мобільних системах попередніх поколінь.

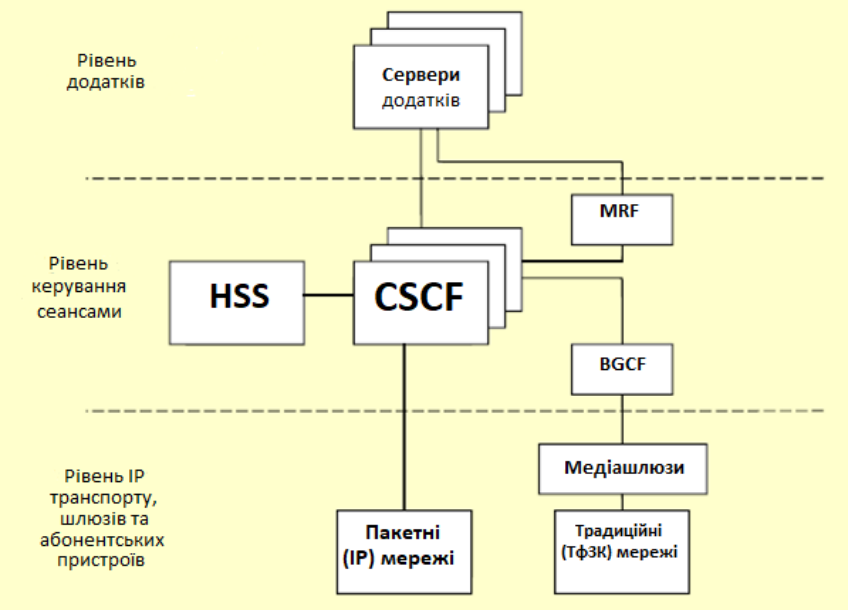

Рисунок 4.1.1 Місце HSS в архітектурі IMS

Функції, що виконує HSS, показані в загальному вигляді на рисунку 4.1.2.

Мережа може містити більше одного HSS в тому випадку, якщо кількість абонентів занадто велика, щоб підтримуватися одним HSS. Така мережа, поряд з кількома HSS, повинна буде мати в своєму складі функцію SLF, що представляє собою просту базу даних, яка зберігає дані про відповідність інформації HSS адресами користувачів. Вузол, який передав до SLF запит з адресою користувача, отримує від неї відомості про той HSS, який містить інформацію про цього користувача. [3]

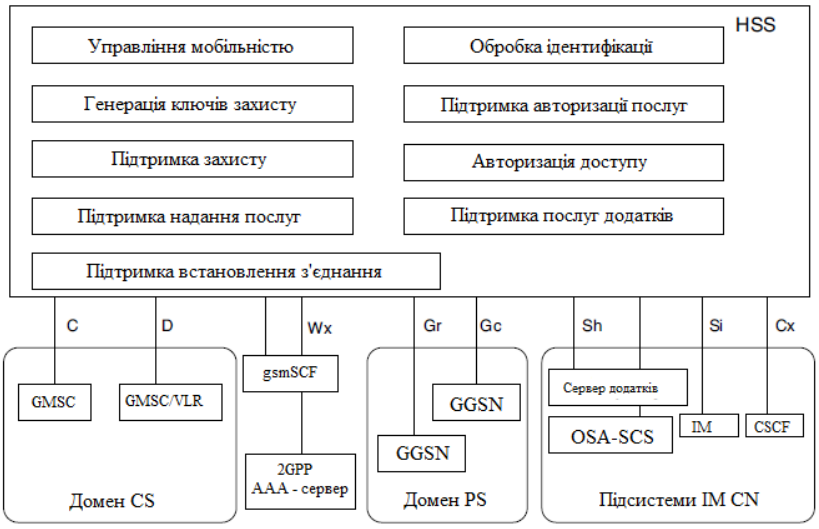

Рисунок 4.1.2 Логічні функції HSS

4.2 Платформа RabbitMQ

RabbitMQ - один з найпопулярніших брокерів повідомлень з відкритим вихідним кодом, який має десятки тисяч користувачів, котрий також використовується в усьому світі як для невеликих стартапів, так і в великих підприємствах. Сервіс RabbitMQ є легким і простим в розгортанні як на обладнанні його користувачів, так і в хмарних сервісу.

RabbitMQ працює з багатьма операційними системами і хмарними середовищами, а також надає широкий спектр інструментів для розробників і для більшості популярних мов.

Особливості RabbitMQ:

- Асинхронний обмін повідомленнями - підтримує декілька протоколів обміну повідомленнями, черги повідомлень, підтвердження доставки, гнучку маршрутизацію в черзі, кілька типів обміну;
- Широкі можливості для розробників - включає можливість розробки додатків на великій кількості мов програмування, таких як: Java, .NET, PHP, Python, JavaScript, Ruby, Go, C# і багатьох інших;
- Розподілене розгортання - розгортання у вигляді кластерів для забезпечення високої доступності та пропускної здатності по географічно рознесених регіонах;

- Підприємливість і готовність до роботи в хмарі - змінна аутентифікація, авторизація, підтримка TLS (Transport Layer Security) і LDAP (Lightweight Directory Access Protocol). Легкість і простота в розгортанні в публічних і приватних хмарах;
- Інструменти та плагіни - різноманітний набір інструментів і плагінів, що підтримують безперервну інтеграцію, операційні показники і інтеграцію з іншими корпоративними системами. Гнучкий підхід до впровадження плагінів для розширення функціональності RabbitMQ;
- Управління та моніторинг - HTTP-API, інструмент командного рядка і інтерфейс користувача, що призначений для управління і моніторингу роботи RabbitMQ. [19]

RabbitMQ - програмний брокер повідомлень, обмін якими, між різними компонентами мережі, реалізований на основі стандарту AMQP (Advanced Message Queuing Protocol) і системи Open Telecom Platform, написаний на мові Erlang і базується на базі СУБД Mnesia, яка, в свою чергу, також написана на Erlang. Mnesia - це розподілена СУБД реального часу, за своєю суттю, використовується для вбудованих рішень і тим самим схожа на Berkeley DB. Він призначений для передачі даних (повідомлень) між декількома сервісами: один сервіс додається в чергу повідомлення, інший - отримує це повідомлення.

Якщо провести аналогію між RabbitMQ і СУБД (Система управління базами даних), то:

- Віртуальний хост в RabbitMQ це як окрема БД (База даних) в СУБД
- Кожна точка обміну - таблиця
- Кожне повідомлення - рядок в таблиці

AMQP протокол має три базових поняття:

- Точка обміну (exchange)
- Черга (queue)
- Зв'язок, або маршрут (routing key)

Якщо точки обміну за своєю природою призначені для публікації повідомлень, то черги призначені для прийому повідомлень. Між точкою

обміну і чергою встановлюється зв'язок (Bind - прив'язка) через ключ маршрутизації (routing_key). [20]

Обмін повідомленнями здійснюється в межах однієї точки обміну, в якій визначені зв'язки, що є своєрідними маршрутами, за якими йдуть повідомлення, що потрапили в цю точку обміну. Кожен маршрут пов'язує точку обміну з однією або декількома чергами. Програмне забезпечення, що реалізує описані дії, називають AMQP-сервером або брокером. Вузли, що поміщають повідомлення в точку обміну і отримують їх з черг, називаються AMQP-клієнтами. Іншими словами, AMQP-сервер надає шину обміну даними, а AMQP-клієнти використовують цю шину для обміну повідомленнями між собою.

Клієнти можуть створювати нові точки обміну і черги:

Exchange.DeclareTYPE

Queue.Declare

Процес переміщення даних на шину називається публікацією. Коли AMQP-клієнт публікує повідомлення на шину, він задає ім'я точки обміну і ключ маршрутизації (routing_key).

Message.PublishTOWITH

До цього моменту, один з клієнтів повинен здійснити зв'язку, вказавши ім'я точки обміну, черги і маршрут, що їй відповідає:

Queue.BindTOWITH

В результаті, повідомлення буде доставлено в чергу. Але на цьому ще не все, так як шлях повідомлення чергою не закінчується. Один з AMQP-клієнтів підписується на отримання повідомлень, які потрапили в задану чергу. Після підписки на чергу, всі повідомлення, що потрапили в неї, будуть пересилатися з підписаним процесом. Якщо на одну чергу підписано кілька процесів, повідомлення будуть розподілятися по підписниками методом round-robin.

Черги можуть бути:

- Ті що самовидаляються (AMQP_AUTODELETE) - видаляються якщо вони порожні і не використовуються (немає клієнтських з'єднань);

- Ті що зберігаються (AMQP_DURABLE) - при перезапуску брокера черги зберігають дані;
- Ексклюзивні (AMQP_EXCLUSIVE) - розраховані тільки на одне з'єднання;
- Пасивні (AMQP_PASSIVE) - ініціювання йде з боку клієнта.

Якщо жоден параметр не заданий, то за замовчуванням черга оголошується як то, що самовидаляється (AMQP_AUTODELETE).

Прив'язка черги до точки обміну здійснюється через ключ маршрутизації. Ключ може бути простий або складний. Для складних ключів використовуються патерни, або ж шаблони. [20]

Точки обміну бувають трьох типів:

- fanout - повідомлення передається в усі прив'язані до неї черги. Не використовує ключів, тому значення ключа вказується формально, задається порожній рядок;
- direct - повідомлення передається в чергу з ім'ям, що збігається з ключем маршрутизації (routing key) (ключ маршрутизації вказується при відправці повідомлення). Використовуються тільки прості ключі;
- topic - щось середнє між fanout і exchange, повідомлення передається в черги, для яких збігається маска на ключ маршрутизації (патерн), наприклад, app.notification.sms.* - в чергу будуть доставлені всі повідомлення, відправлені з ключами, що починаються на app.notification.sms.

Якщо черга прив'язана до пунктів обміну і в цих точках публікуються повідомлення з ключем прив'язки, то ці повідомлення перенаправляються в відповідну прив'язці чергу.

Зчитувати з черги повідомлення можна двома способами:

- асинхронно, через метод Get ()
- синхронно, через метод Consume ()

Метод Get () зчитує з черги одне повідомлення. При читанні повідомлення, в заголовному фреймі передається інформація скільки ще повідомлень залишилося в черзі.

У методу Get () може бути параметр AMQP_NOACK, який «говорить» брокеру, що це повідомлення не позначати як «прочитане».

Спільно з параметром AMQP_NOACK, використовується метод Ack (), який підтверджує, що повідомлення доставлено. Вся інформація про повідомленні інкапсулюється в об'єкті даних AMQPMessage.

У об'єкту «повідомлення» є методи для доступу полів, назви говорять самі за себе: getMessage(), getExchange(), getRoutingKey(), get MessageCount(). Слід загострити увагу на методах getConsumerTag() і getDeliveryTag().

Тег підписника (consumer_tag) - це індивідуальний неповторний рядок, призначається або при публікації, або автоматично брокером, схоже на ім'я сесії. Наприклад, відмовитися від підписки можна, надіславши команду Cancel з передачею в параметрі даних consumer_tag:

AMQPQueu::Cancel(m->getConsumerTag())

Тег доставки (delivery_tag) - це числове значення, рівне лічильнику доставлених повідомлень даної сесії, для першого повідомлення тег доставки дорівнює - 1, для другого - 2, для третього - 3 і так далі. Для підтвердження прийому повідомлення, необхідно викликати метод Ack (delivery_tag), де змінна delivery_tag має значення тега доставки.

На відміну від методу Get, метод Consume має синхронну схему прийому, тому тут використовується модель подій. Перед використанням методу підписка (Consume) необхідно додати подію AMQP_MESSAGE. [20]

4.3 Моделювання процесу прийому та обробки заявок функціональним обладнанням мережі

В даній моделі проводиться моделювання процесу надходження та обробки заявок про надання певних телекомунікаційних послуг, що надходять від користувачів мережі зв’язку на HSS.

Запити, що надходять від користувачів модельованої мережі:

- надання доступу до мережі Інтернет (internet_service);

- здійснення відео дзвінка (video_call);
- здійснення голосового дзвінка (voice_call).

Наведені типи заявок було обрано для проведення експериментального дослідження керуючись різницею в кількості їх надходження до HSS за 1 секунду. Таким чином, заявок на здійснення відео дзвінка буде найменше, що пов’язано з непопулярністю послуги, тоді як запитів на надання доступу до мережі Інтернет буде значно більша кількість.

Експериментальна модель складається з п’яти блоків, що наведено на рисунку 4.3.1.

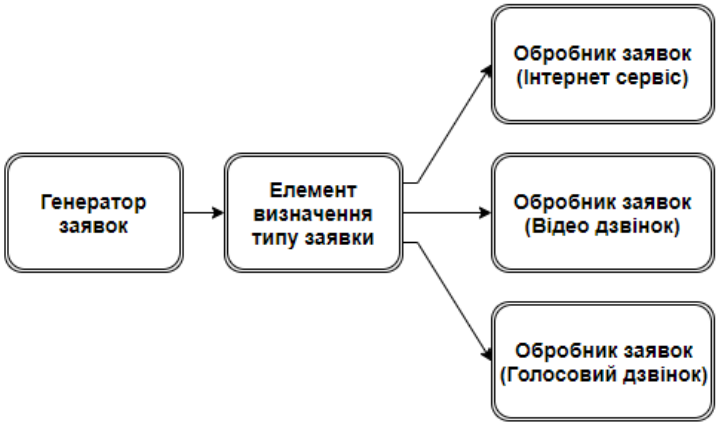

Рисунок 4.3.1 Схема експериментальної моделі процесу прийому та обробки заявок

Генератор заявок формує потік запитів трьох типів з різною інтенсивністю, згідно з вхідними даних експерименту.

Елемент визначення типу заявок виконує роль аналізатору запитів, що на нього надходять. Таким чином, обробивши повідомлення він визначає його подальший шлях, тобто якому з обробників воно призначене.

Обробники заявок, яких у моделі наведено три типи, займаються обробкою запитів, які надходять на них від елементу визначення типу заявки. Слід зауважити, що один обробник може працювати лише з заявками свого типу. Даний елемент є функціональною частиною системи SEDA , що містить в собі певну кількість етапів SEDA. Кількість стадій або ж етапів може

регулюватись як вручну, так і автоматично, в залежності від ресурсів, що виділяються на систему, інтенсивності надходження заявок та їх кількості у черзі на обробку. Самі ж черги запитів, після проходження елементу визначення типу заявок, виникають на вході обробників заявок, SEDA контролер пересилає повідомлення з входу обробника запитів і передає їх на вхід етапу SEDA, з чого можна зробити висновок, що експериментальна модель має ієрархічну структуру.

Для початку проведення дослідження необхідно визначитись з вхідними даними експерименту. Для початку роботи системи базова мережа етапів SEDA повинна складатись мінімум з одного етапу кожного типу. Значення кількості стадій може варіюватись в ході дослідження, в залежності від потреби. В той же час воно не може бути нульовим, так як у такому випадку заявки, що надходять, не будуть оброблятись.

В наведеній моделі реалізується необмежена черга без втрат, яку слід вважати квазі необмеженою, так як ресурси персонального комп’ютера на якому проводиться дослідження обмежені. Графіки та показники продуктивності, що наводяться далі, динамічні, тому що процес роботи системи відбувається в реальному часі.

Таблиця 4.1 Вхідні дані моделювання

Тип параметру	Назва параметру	Значення параметру
Інтенсивність надходження заявок	Internet_service	37/c
	Video_call	6/c
	Voice_call	17/c
Кількість обробників	Handler_internet_service	1
	Handler_video_call	1
	Handler_voice_call	1

Моделювання буде проводитися в 4 режимах:

1. Режим накопичення, в якому, згідно початкових даних, генерується безкінечна кількість заявок, що накопчуються у черзі. У компоненті мережі знаходиться по одному обробнику заявок, що працюють в режимі простою, тобто запити, що поступають на них не оброблюються, формується черга повідомлень готових до обробки.

2. Базовий режим, в якому вмикається по одному SEDA етапу у кожному з обробників заявок.

3. Режим обробки, в якому проводиться підбір кількості SEDA етапів для забезпечення обробки необхідного обсягу заявок відносно інтенсивності їх надходження з черги.

4. Режим відсутності загальної черги, в якому робота компоненту мережі відбувається з залученням всіх доступних йому ресурсів, черга не формується.

1) Режим накопичення.

Підготувавши віртуальний стенд моделі компоненту нашої мережі розпочинаємо його роботу. Слід відзначити, що на графіках, які в подальшому будуть наведені, можливі викривлення, що не зовсім відповідають реальній роботі змодельованої системи. Це обумовлено як апаратними можливостями персонального комп’ютеру, на якому проводиться експеримент, а також особливостями відображення графічних показників сервером RabbitMQ.

На рисунку 4.3.2 зображено роботу моделі у режимі накопичення, де обробники знаходяться у стані простою. На даному рисунку присутні наступні значення: Ready – кількість заявок готових до обробки; Unacked – кількість помилкових заявок, які виникають, коли заявка одночасно генерується та потрапляє у чергу. Так заявки такого типу перенаправляються на повторну обробку, а помилка зникає, таким чином всі заявки буде оброблено; Total – загальна кількість заявок у черзі.

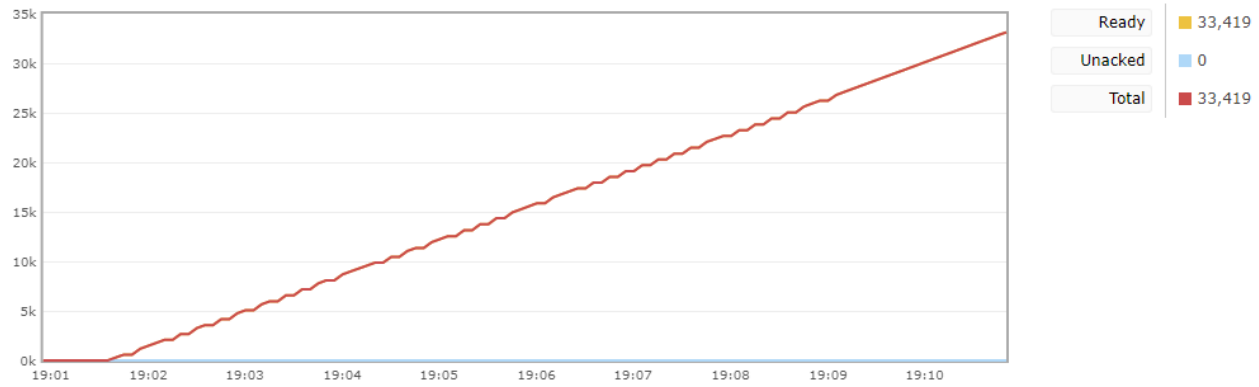

Рисунок 4.3.2 Накопичення заявок у загальній черзі при роботі в режимі накопичення

Рисунок 4.3.3 ілюструє дані виміру продуктивності усієї моделі в цілому, тобто її зовнішньої та внутрішньої частини. Значення, що необхідні для експерименту:

- Publish – інтенсивність надходження заявок до компоненту мережі, або ж кількість згенерованих заявок;
- Consumer ack – кількісний показник готовності обробників заявок на обробку певного обсягу запитів;
- Deliver – інтенсивність обробки заявок обробником.

На подальших рисунках даного експериментального дослідження, не тільки тих, що стосуються даного режиму роботи системи, буде змінюватись точка проведення вимірювань показників різних компонентів схеми та кількість робочих елементів, тобто етапів SEDA, в обробниках заявок.

Рисунок 4.3.3 Інтенсивність надходження, готовності та обробки заявок у загальній черзі при роботі в режимі накопичення

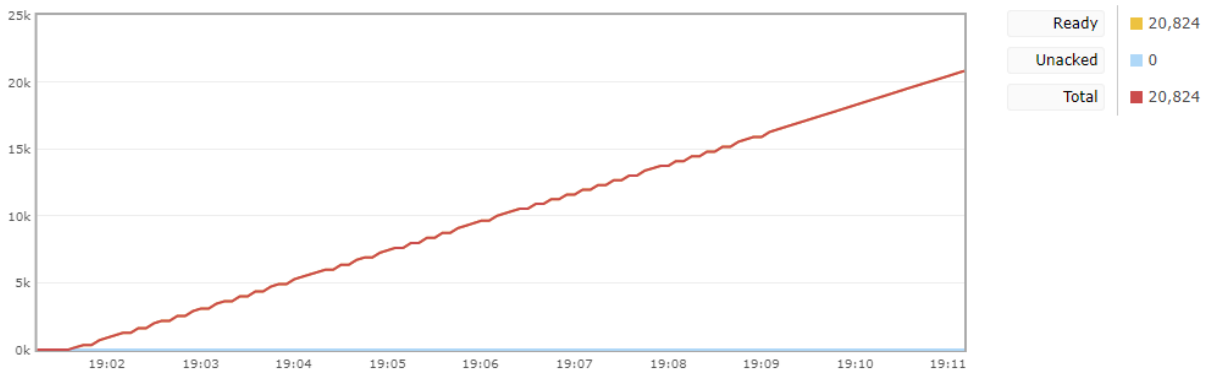

Рисунок 4.3.4 Накопичення заявок у черзі обробника Handler_internet_service при роботі у режимі накопичення

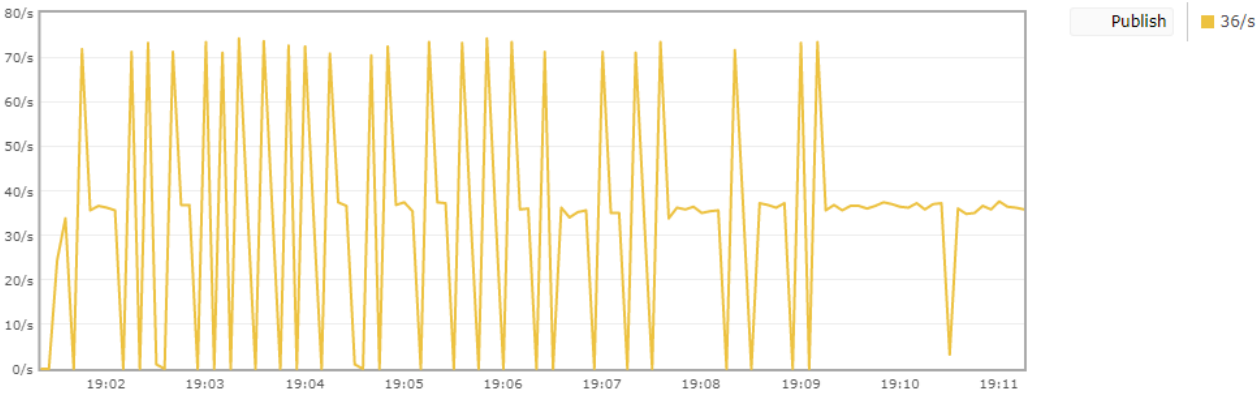

Рисунок 4.3.5 Інтенсивність надходження, готовності та обробки заявок у черзі обробника Handler_internet_service при роботі в режимі накопичення

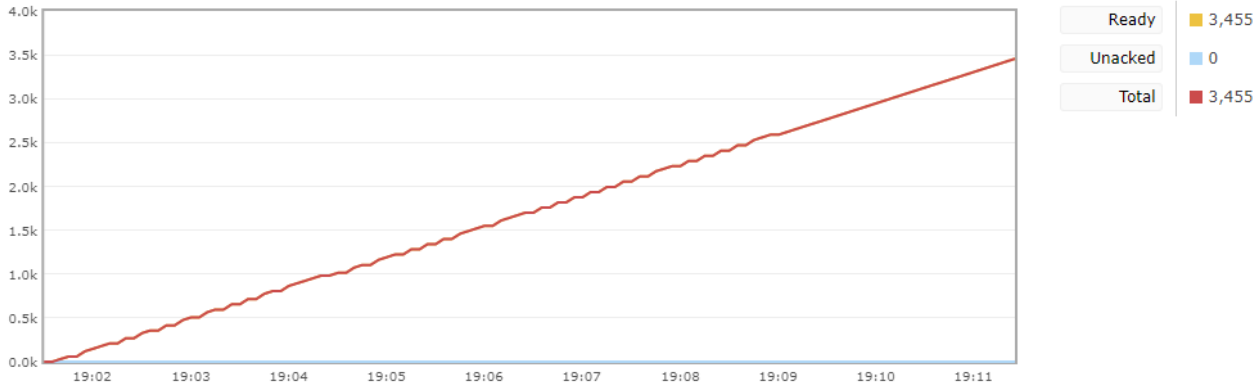

Рисунок 4.3.6 Накопичення заявок у черзі обробника Handler_video_call при роботі в режимі накопичення

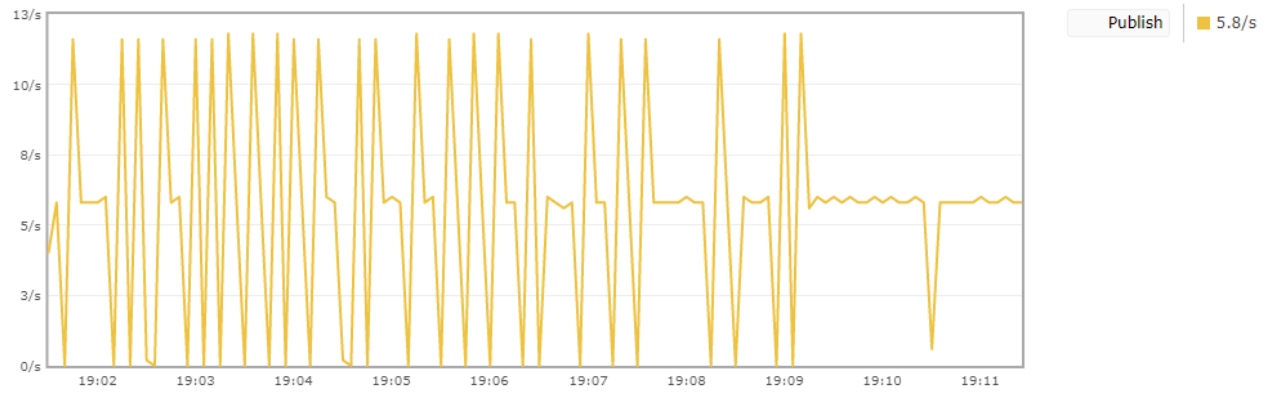

Рисунок 4.3.7 Інтенсивність надходження, готовності та обробки заявок у черзі обробника Handler_video_call при роботі в режимі накопичення

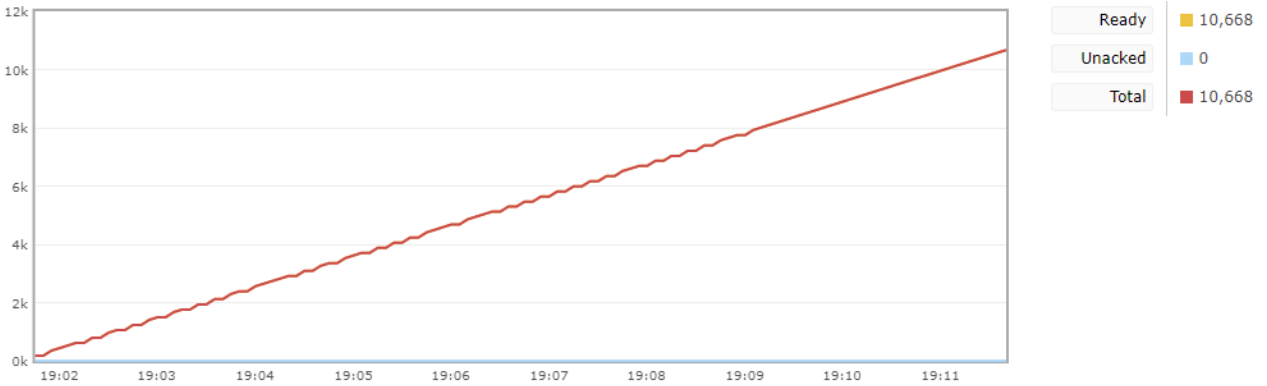

Рисунок 4.3.8 Накопичення заявок у черзі обробника Handler_voice_call при роботі в режимі накопичення

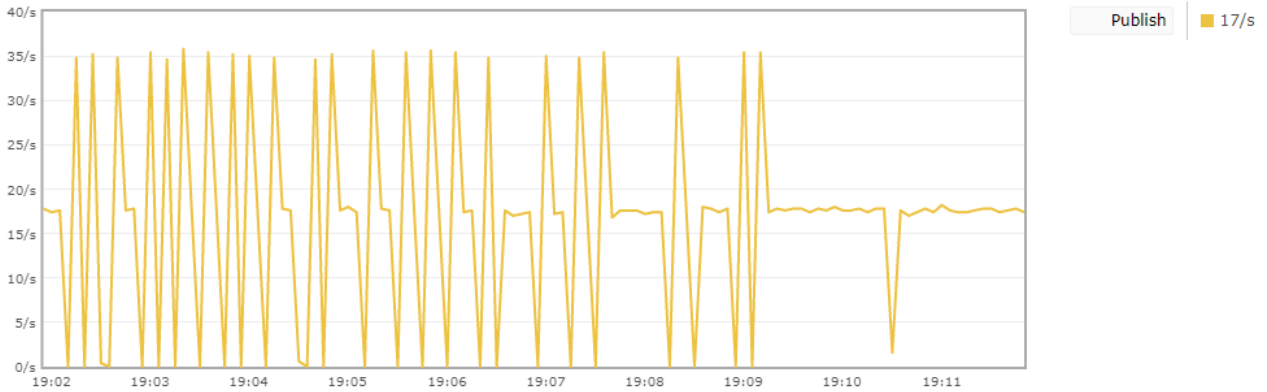

Рисунок 4.3.9 Інтенсивність надходження, готовності та обробки заявок у черзі обробника Handler_voice_call при роботі в режимі накопичення

У таблиці 4.2 наведено зведену інформацію про утворені черги на вході кожного з обробників системи.

Таблиця 4.2 Зведені дані по трьом чергам, при роботі в режимі накопичення

Overview			Messages			Message rates		
Name	**Consumers**	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**
internet_service	0	running	25,873	0	25,873	36/s		
video_call	0	running	4,172	0	4,172	5.8/s		
voice_call	0	running	12,610	0	12,610	17/s		

Згідно даної таблиці можна зробити висновок, що змодельована система генерує необмежену чергу заявок без втрат, обробка запитів не відбувається, так як кількість активних етапів SEDA рівна 0.

2) Базовий режим.

При моделюванні роботи даного режиму в певний момент часу в досліджувану систему на кожен обробник заявок вводиться по одному активному етапу SEDA. Даний процес можна спостерігати на графічних ілюстраціях, що наведено далі. Невеликий перегин на графіках кількості повідомлень у черзі характеризує зменшення інтенсивності потрапляння заявок до черги, що обумовлено початком обробки певного обсягу запитів. Сам початок обробки повідомлень можна спостерігати на рисунках інтенсивності надходження, готовності та обробки заявок.

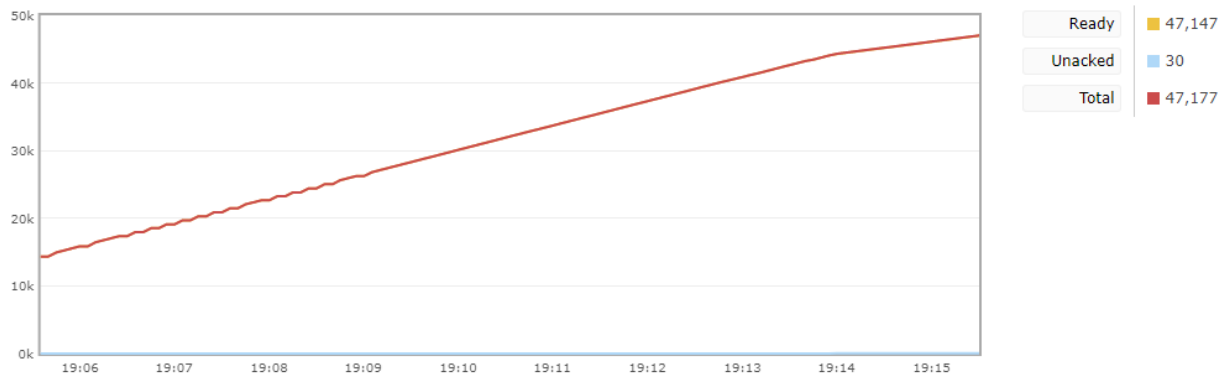

Рисунок 4.3.10 Накопичення заявок у загальній черзі при роботі в базовому режимі

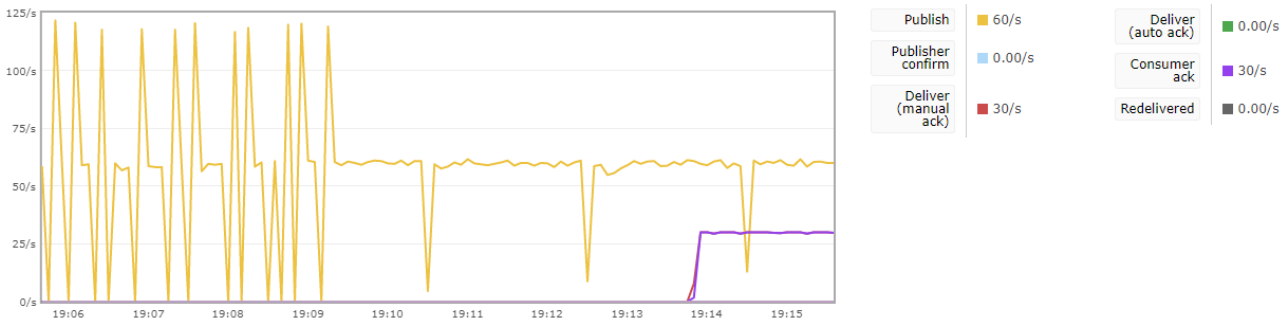

Рисунок 4.3.11 Інтенсивність надходження, готовності та обробки заявок у загальній черзі при роботі в базовому режимі

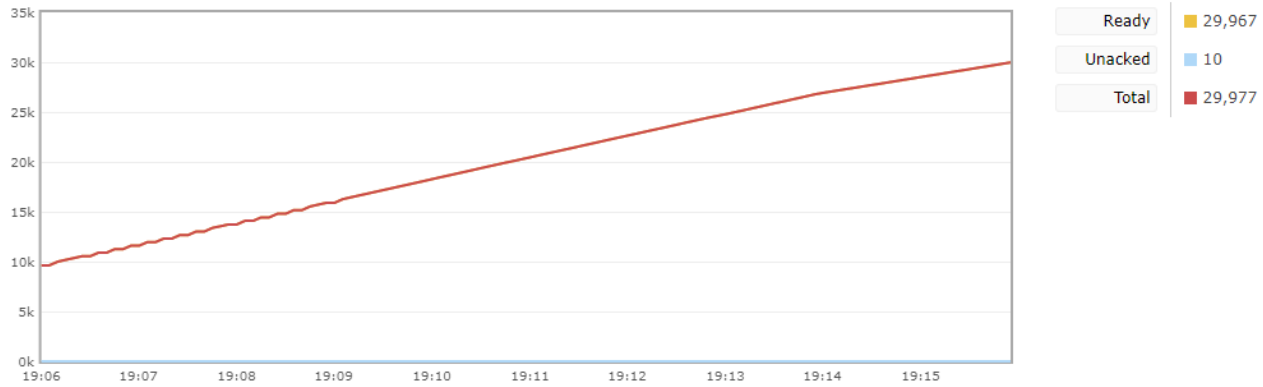

Рисунок 4.3.12 Накопичення заявок у черзі обробника Handler_internet_service при роботі в базовому режимі

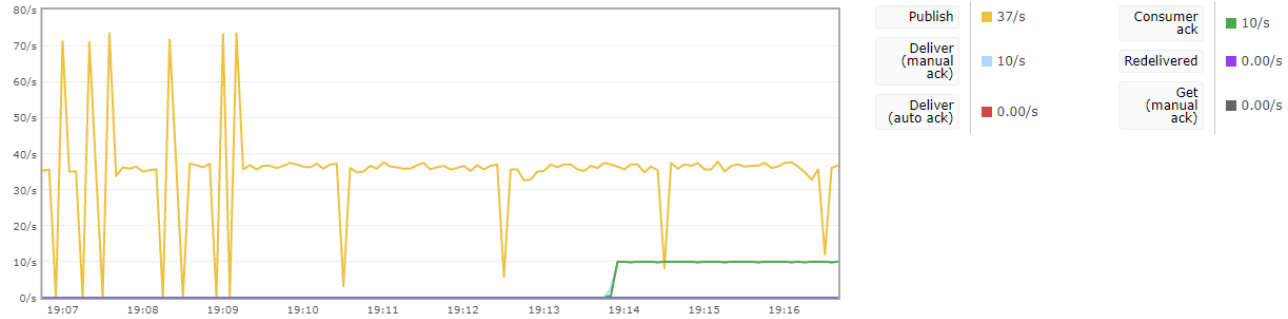

Рисунок 4.3.13 Інтенсивність надходження, готовності та обробки заявок у черзі обробника Handler_internet_service при роботі в базовому режимі

Таблиця 4.3 Значення параметрів обробника Handler_internet_service при роботі в базовому режимі

Features	State	running		Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Policy	Consumers	1	Messages ?	31,555	31,545	10	31,555	0	0
Operator policy	Consumer utilisation ?	0%	Message body bytes ?	401kiB	400kiB	130iB	401kiB	0iB	0iB
Effective policy definition			Process memory ?	31MiB					

Згідно таблиці 4.3, можна оцінити об’єм фізичної пам’яті, що виділяється на даний обробник (Process memory), об’єм оброблених даних (Message body bytes), кількість етапів SEDA (Consumers) та коефіцієнт використання обробника (Consumer utilization).

Коефіцієнт використання обробника рівний відношенню часу приходу наступної заявки до часу очікування заявки на обробку. Таким чином цей параметр показує готовність обробника, приймати нові заявки. Значення цього параметру менше 100% означає, що додаткове виділення ресурсів, тобто підвищення інтенсивності обробки запитів, підвищить продуктивність роботи обробника повідомлень. Так як в сервері RabbitMQ реалізована система черг по алгоритму LIFO (Last In, First Out), то значення Consumer utilization рівне 0% абсолютно логічне.

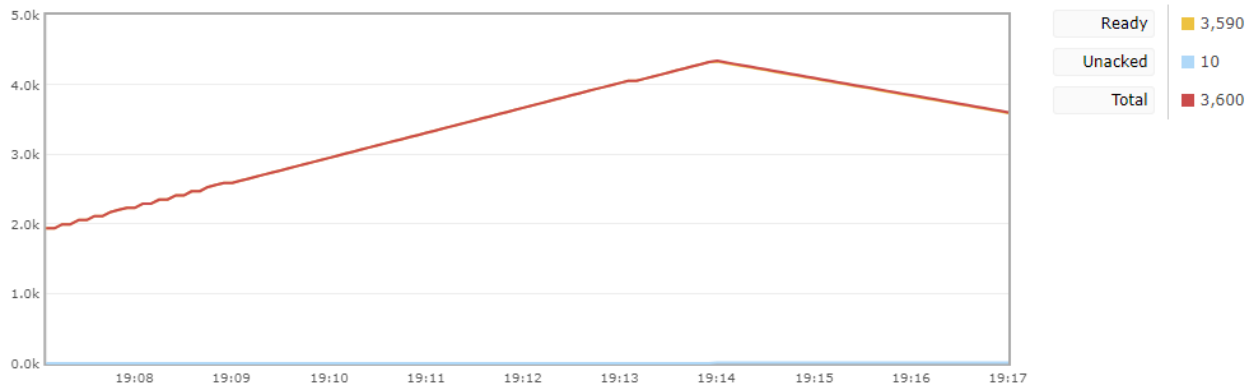

Рисунок 4.3.14 Накопичення заявок у черзі обробника Handler_video_call при роботі в базовому режимі

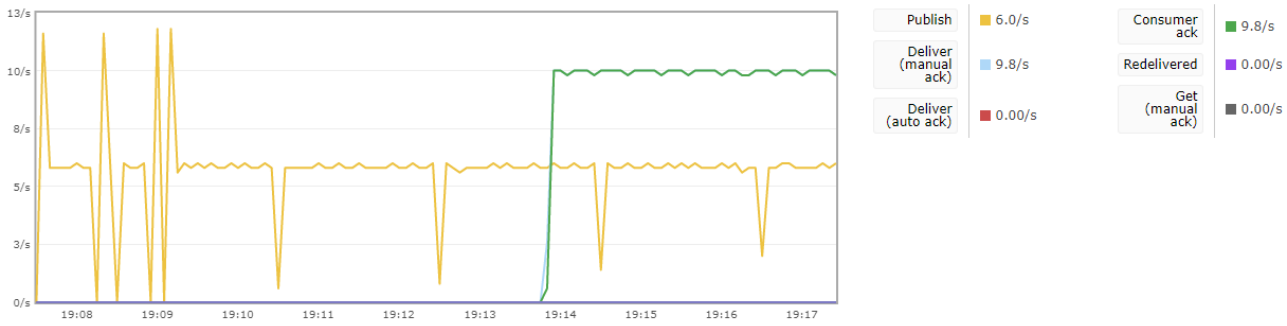

Рисунок 4.3.15 Інтенсивність надходження, готовності та обробки заявок у черзі обробника Handler_video_call при роботі в базовому режимі

Таблиця 4.4 Значення параметрів обробника Handler_video_call при роботі в базовому режимі

Features	State	running		Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Policy	Consumers	1	Messages ?	3,436	3,426	10	3,436	0	0
Operator policy	Consumer utilisation ?	0%	Message body bytes ?	44kiB	43kiB	130iB	44kiB	0iB	0iB
Effective policy definition			Process memory ?	7.3MiB					

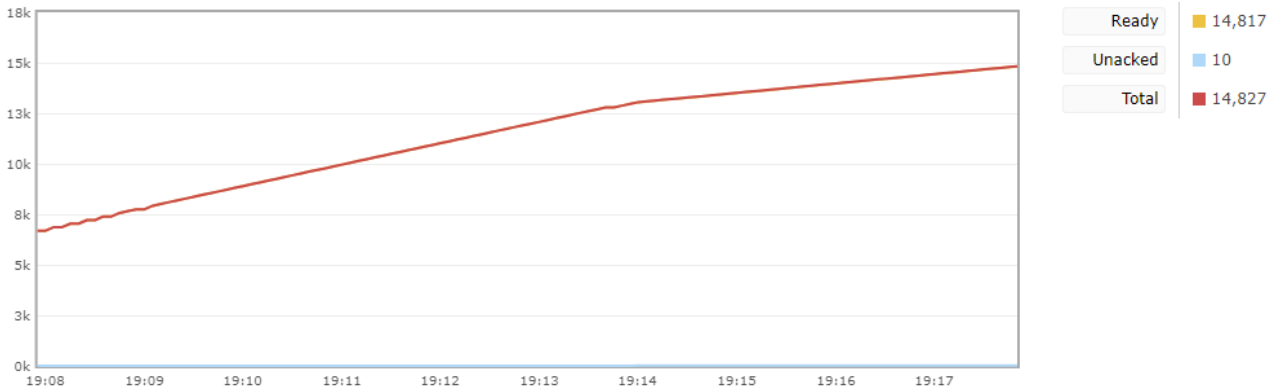

Рисунок 4.3.16 Накопичення заявок у черзі обробника Handler_voice_call при роботі в базовому режимі

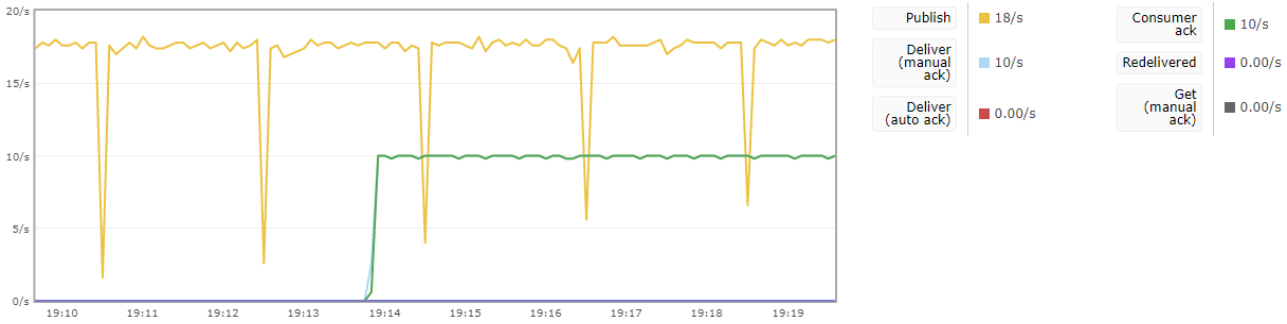

Рисунок 4.3.17 Інтенсивність надходження, готовності та обробки заявок у черзі обробника Handler_voice_call при роботі в базовому режимі

Таблиця 4.5 Значення параметрів обробника Handler_voice_call при роботі в базовому режимі

Features	State	running		Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Policy	Consumers	1	Messages ?	15,768	15,758	10	15,768	0	0
Operator policy	Consumer utilisation ?	0%	Message body bytes ?	200kiB	200kiB	130iB	200kiB	0iB	0iB
Effective policy definition			Process memory ?	18MiB					

Таблиця 4.6 Зведені дані по трьом чергам, при роботі в базовому режимі

Overview			Messages			Message rates		
Name	**Consumers**	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**
internet_service	1	running	38,201	10	38,211	37/s	10/s	10/s
video_call	1	running	2,584	10	2,594	5.8/s	10/s	10/s
voice_call	1	running	16,331	10	16,341	18/s	10/s	10/s

3) Режим обробки.

Виходячи з початкових даних та обмежень в апаратній частині персонального комп'ютера, на якому проводиться моделювання, для проведення експериментального дослідження було встановлено, що оптимальною кількістю етапів SEDA буде 7, тому розподіл відбувся наступним чином:

- Handler_internet_service – 4;
- Handler_video_call – 1;
- Handler_voice_call – 2.

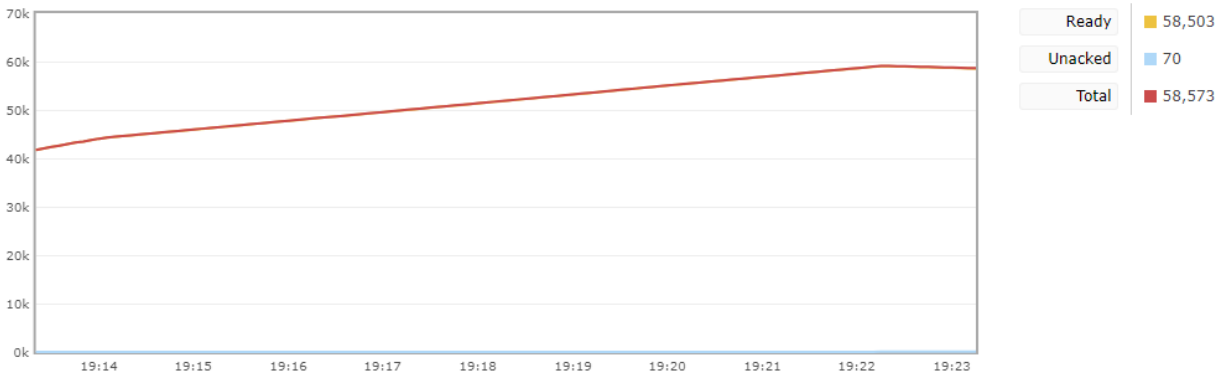

Рисунок 4.3.18 Накопичення заявок у загальній черзі при роботі в режимі обробки

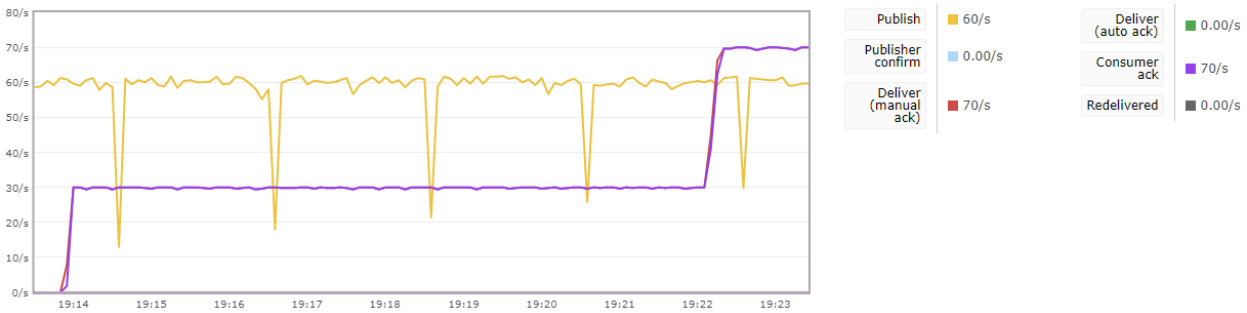

Рисунок 4.3.19 Інтенсивність надходження, готовності та обробки заявок у загальній черзі при роботі в режимі обробки

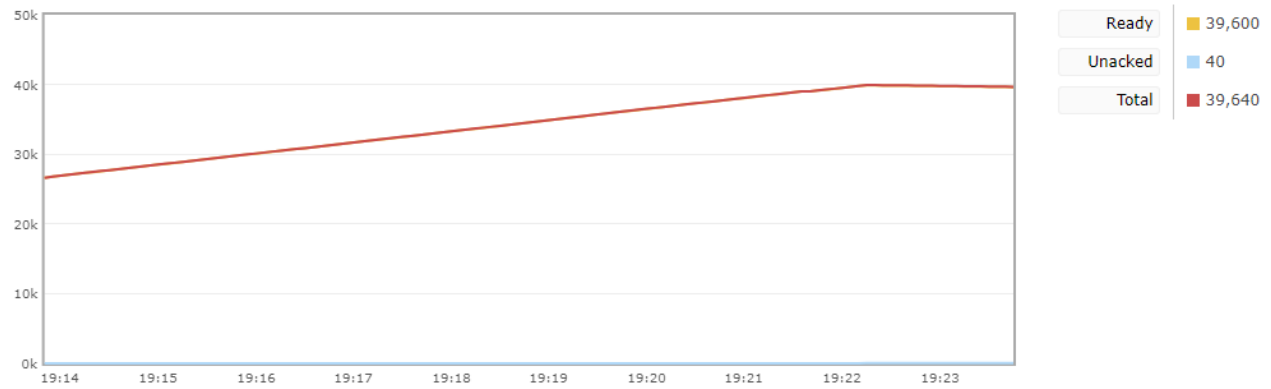

Рисунок 4.3.20 Накопичення заявок у черзі обробника Handler_internet_service при роботі в режимі обробки

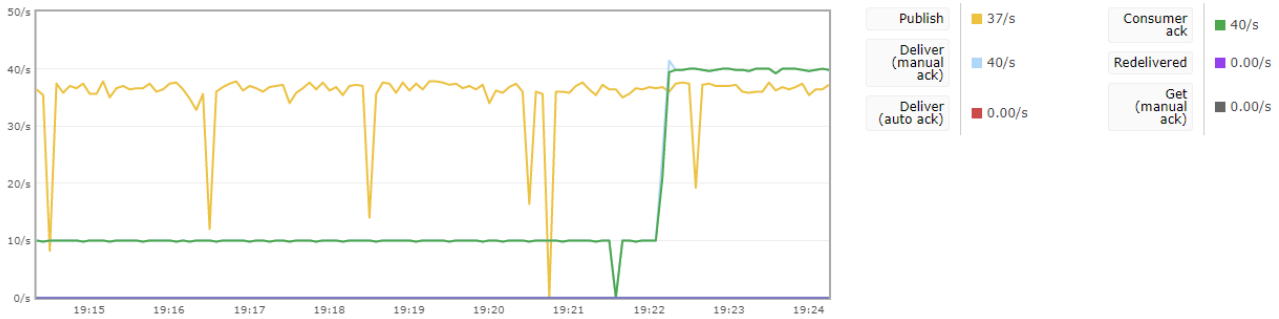

Рисунок 4.3.21 Інтенсивність надходження, готовності та обробки заявок у черзі обробника Handler_internet_service при роботі в режимі обробки

Таблиця 4.7 Значення параметрів обробника Handler_internet_service при роботі в базовому режимі

Features	State	running		Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Policy	Consumers	4	Messages ?	39,511	39,471	40	39,511	0	0
Operator policy	Consumer utilisation ?	0%	Message body bytes ?	502kiB	501kiB	520iB	502kiB	0iB	0iB
Effective policy definition			Process memory ?	54MiB					

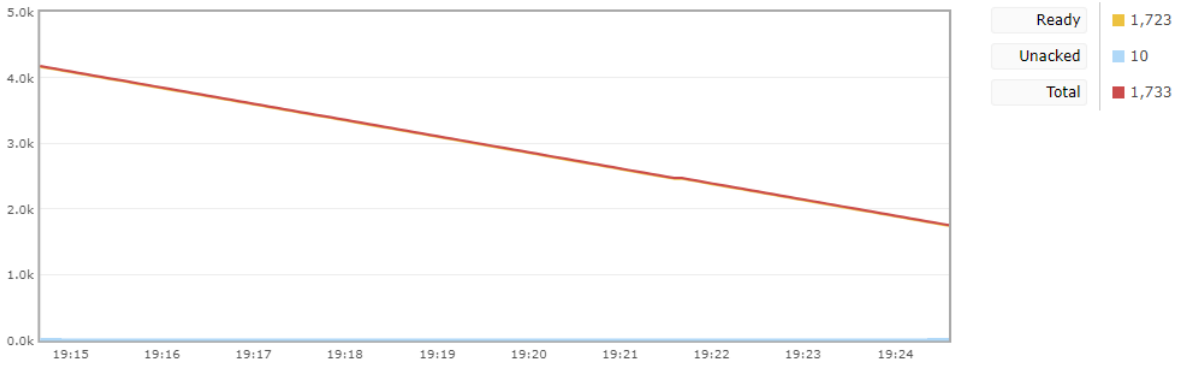

Рисунок 4.3.22 Накопичення заявок у черзі обробника Handler_video_call при роботі в режимі обробки

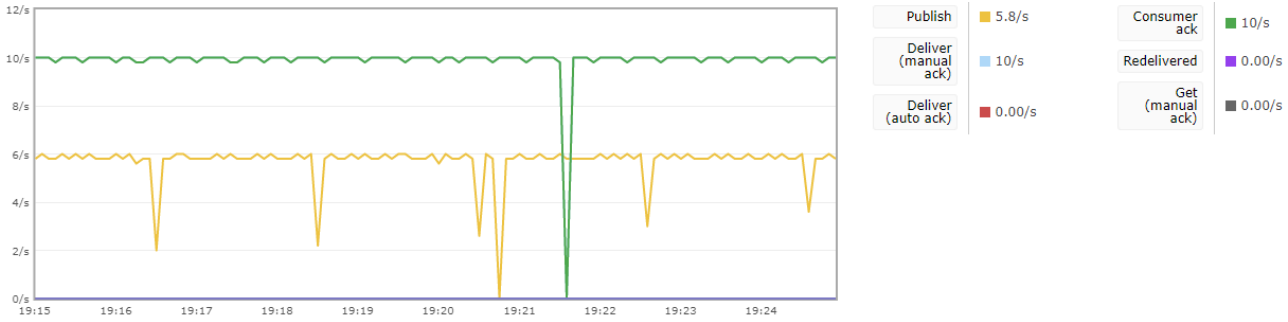

Рисунок 4.3.23 Інтенсивність надходження, готовності та обробки заявок у черзі обробника Handler_internet_service при роботі в режимі обробки

Таблиця 4.8 Значення параметрів обробника Handler_video_call при роботі в базовому режимі

Features	State	running		Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Policy	Consumers	1	Messages ?	1,609	1,599	10	1,609	0	0
Operator policy	Consumer utilisation ?	0%	Message body bytes ?	20kiB	20kiB	130iB	20kiB	0iB	0iB
Effective policy definition			Process memory ?	7.3MiB					

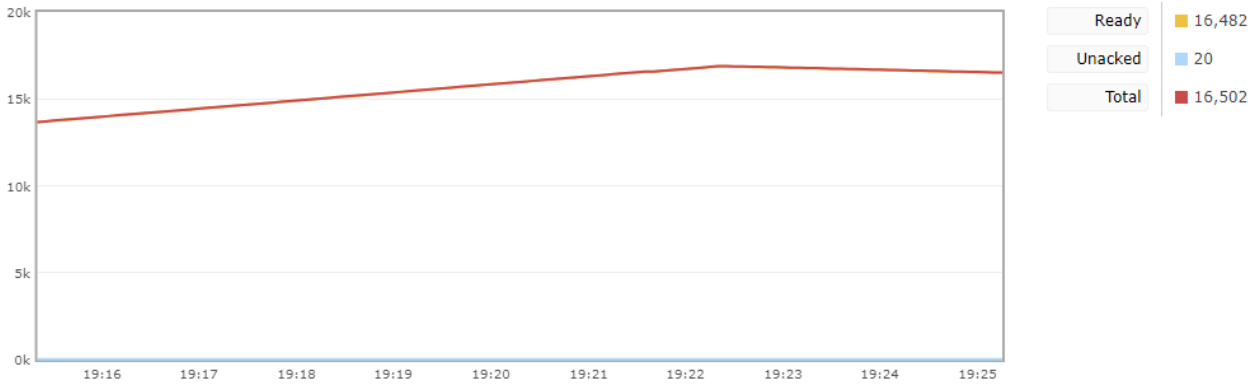

Рисунок 4.3.24 Накопичення заявок у черзі обробника Handler_voice_call при роботі в режимі обробки

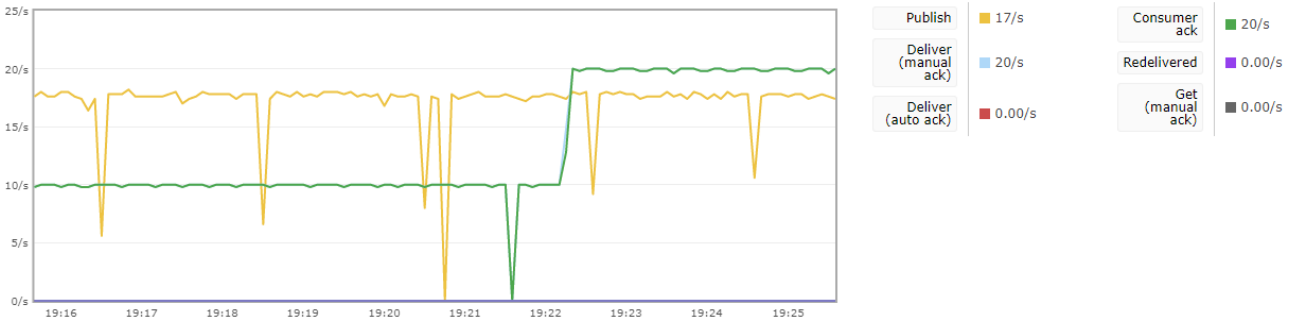

Рисунок 4.3.25 Інтенсивність надходження, готовності та обробки заявок у черзі обробника Handler_voice_call при роботі в режимі обробки

Таблиця 4.9 Значення параметрів обробника Handler_voice_call в базовому режимі

Features	State	running		Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Policy	Consumers	2	Messages ?	16,421	16,401	20	16,421	0	0
Operator policy	Consumer utilisation ?	0%	Message body bytes ?	208kiB	208kiB	260iB	208kiB	0iB	0iB
Effective policy definition			Process memory ?	18MiB					

Таблиця 4.10 Зведені дані по трьом чергам, при роботі в режимі обробки

Overview			Messages			Message rates		
Name	**Consumers**	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**
internet_service	4	running	39,064	40	39,104	37/s	40/s	40/s
video_call	1	running	1,250	10	1,260	5.8/s	10/s	10/s
voice_call	2	running	16,311	20	16,331	18/s	20/s	20/s

Отже реальна продуктивність кожного обробника подій становить:

- Handler_internet_service – 40 з/с;
- Handler_video_call – 10 з/с;
- Handler_voice_call - 20 з/с.

4) Режим відсутності загальної черги

Робота режиму відсутності загальної характерна тим, що на роботу обробників повідомлень виділяються всі доступні ресурси мережі, таким чином забезпечуючи коефіцієнт використання обробника рівним 100%, тобто всі заявки які поступають на даний елемент оброблюються без затримки. Іншими словами, у даному режимі інтенсивність обробки повідомлень буде більшою ніж інтенсивність їхньої генерації.

В подальшому наводяться графіки роботи даного режиму на яких можна спостерігати, що інтенсивність надходження заявок рівно інтенсивності їхньої обробки. Та дані значення не є показниками реальної продуктивності, так як обробники, при відсутності черги повідомлень, можуть опрацювати всі заявки не використовуючи всі свої ресурси. Насправді ж реальна продуктивність кожного обробника подій була виміряна та наведена у таблиці 4.10 попереднього режиму.

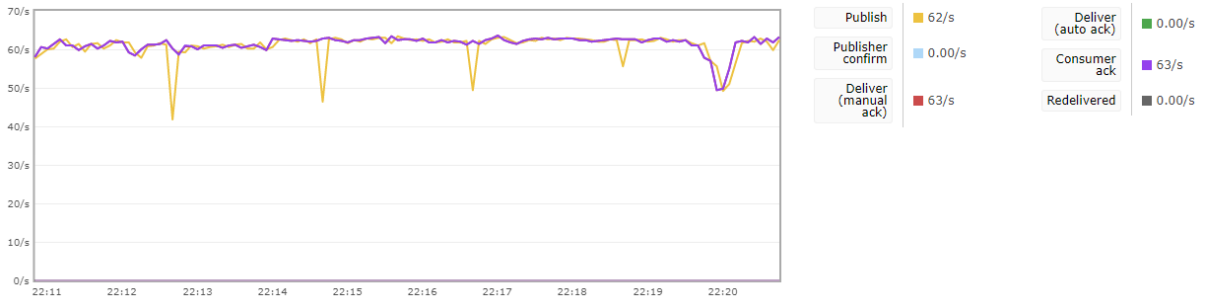

Рисунок 4.3.26 Накопичення заявок у загальній черзі при роботі в режимі відсутності загальної черги

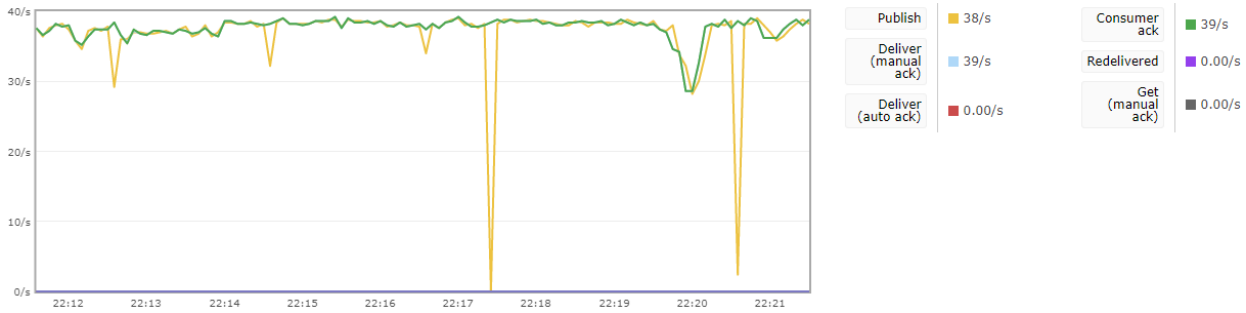

Рисунок 4.3.27 Інтенсивність надходження, отримання та обробки заявок у черзі обробника Handler_internet_service при роботі в режимі відсутності загальної черги

Таблиця 4.11 Параметри першого обробника при роботі в режимі відсутності загальної

Features	State	running		Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Policy	Consumers	4	Messages ?	0	0	0	0	0	0
Operator policy	Consumer utilisation ?	100%	Message body bytes ?	0iB	0iB	0iB	0iB	0iB	0iB
Effective policy definition			Process memory ?	120kiB					

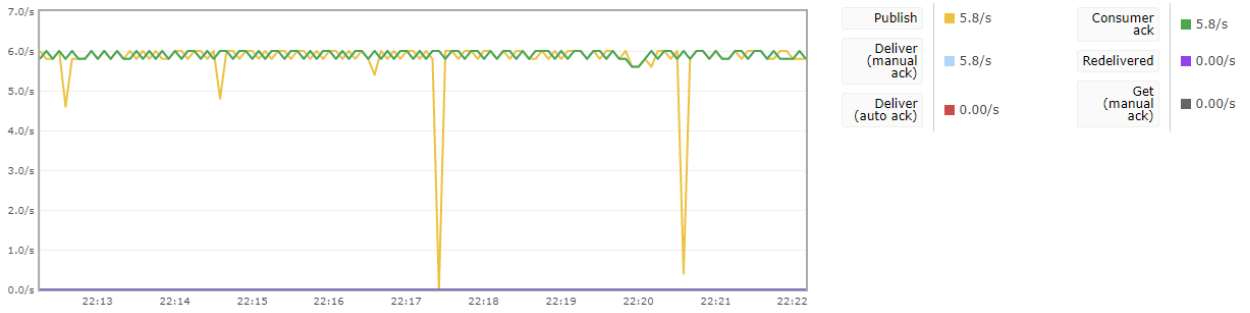

Рисунок 4.3.28 Інтенсивність надходження, отримання та обробки заявок у черзі обробника Handler_video_call при роботі в режимі відсутності загальної черги

Таблиця 4.12 Параметри другого обробника при роботі в режимі відсутності загальної черги

Features	State	running		Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Policy	Consumers	1	Messages ?	0	0	0	0	0	0
Operator policy	Consumer utilisation ?	100%	Message body bytes ?	0iB	0iB	0iB	0iB	0iB	0iB
Effective policy definition			Process memory ?	120kiB					

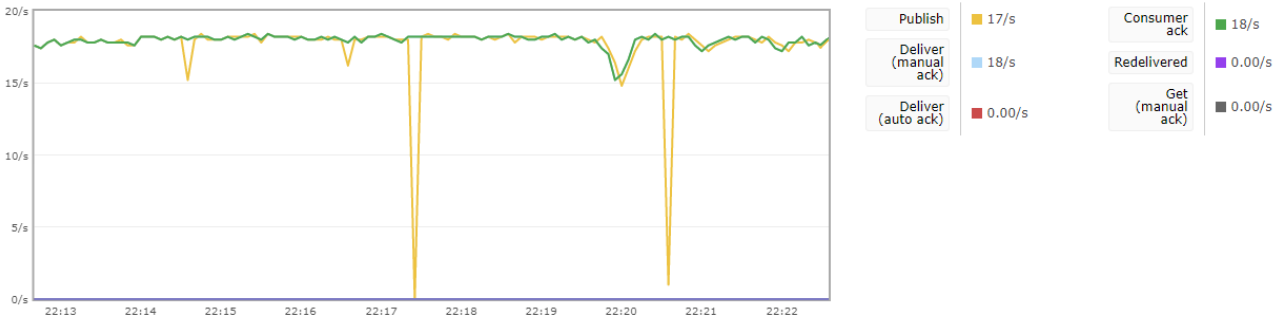

Рисунок 4.3.29 Інтенсивність надходження, отримання та обробки заявок у черзі обробника Handler_voice_call при роботі в режимі відсутності загальної черги

Таблиця 4.13 Параметри третього обробника при роботі в режимі відсутності загальної черги

Features	State	running		Total	Ready	Unacked	In memory	Persistent	Transient, Paged Out
Policy	Consumers	2	Messages ?	0	0	0	0	0	0
Operator policy	Consumer utilisation ?	100%	Message body bytes ?	0iB	0iB	0iB	0iB	0iB	0iB
Effective policy definition			Process memory ?	140kiB					

Таблиця 4.14 Зведені дані по трьом чергам, при роботі в режимі відсутності загальної черги

Overview			Messages			Message rates		
Name	**Consumers**	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**
internet_service	4	running	0	0	0	39/s	39/s	39/s
video_call	1	running	0	0	0	6.0/s	6.0/s	6.0/s
voice_call	2	running	0	0	0	18/s	18/s	18/s

Таблиця 4.14 показує, що отримана кількість SEDA етапів та їх продуктивність, повністю забезпечує систему при сталій інтенсивності вхідного потоку заявок, а також має додаткову ємність при можливих збільшеннях навантажень.

4.4 Висновки до розділу 4

В розділі №4 проводиться експериментальне дослідження одного із компонентів мережі IMS, яку реалізовано у хмарному середовищі, а саме HSS на який надходить велика кількість вхідних заявок про надання послуг для користувачів.

Експериментальну модель розгорнуто на основі серверу RabbitMQ, який має змогу працювати з багатьма операційними системами і хмарними середовищами, а також не є ресурсозатратним, що являється великою перевагою при проведенні дослідження при доступних ресурсах, які обмежені домашнім персональним комп’ютером. Дана платформа має свою систему моніторингу, яка повністю адаптована під архітектуру SEDA. Так проводячи експеримент можна відслідковувати кількість згенерованих заявок, заявок що надійшли в чергу до кожного з блоків програми та тих, які обробляються обробником.

При проведенні дослідження було встановлено, що реальна продуктивність компоненту мережі становить (заявок/сек):

- Handler_internet_service – 39 з/с;
- Handler_video_call – 6 з/с;
- Handler_voice_call - 18 з/с.

Наведені результати затверджують, що обробка вхідного потоку заявок відбувається в повному обсязі. Також за допомогою проведення експериментального дослідження у різних режимах доведено, що при великій кількості вхідних заявок та при відсутності великої черги змодельований компонент мережі встигає обробити всі заявки, що до нього надходять, без простою його роботи.

ВИСНОВКИ

Дослідницька робота, що була проведена в рамках магістерської дисертації, показує, що концепція мультимедійної IP підсистеми грає значну роль у розвитку інфокомунікаційних систем.

Сучасні світові тренди вказують на те, що гостра конкуренція операторів та високий попит на отримання широкого спектру різнопланових послуг приведе до необхідності сервіс провайдерів розширювати сферу свого впливу на ринку інфокомунікацій.

Таким чином, для того щоб утримати своїх абонентів, а також залучати нових, компаніям необхідно шукати нові шляхи вирішення цього питання. Тому вже зараз на ринку відбуваються активні процеси злиття і поглинання. Нерідко консолідуються компанії - провайдери різних типів послуг (фіксована і мобільна телефонія, мобільна телефонія і кабельне телебачення і т.п.). Однак в майбутньому такі процеси приведуть до появи операторів-гігантів інфокомунікацій, в яких буде потужна база користувачів мобільних 4G LTE та стаціонарних мереж, що поставить перед даними компаніями нову проблему – забезпечення конвергенції даних мереж між собою в одну єдину систему, тобто реалізацію концепції FMC. Дана проблема викликала необхідність застосування технології IP Multimedia Subsystem (IMS), яка, дозволяє здійснити інтеграцію мережі NGN та послуг TriplePlay (голос, відео, дані), що у ній надаються, у мобільні мережі, об’єднавши їх в одну і створивши таким чином 4Play (голос, відео, дані + мобільність). Так виникає IMS мережа, яка дозволяє забезпечити надання широкого спектру нових послуг усім користувачам мережі, незалежно від місця їхнього розташування, або ж термінального обладнання яке вони використовують, при цьому забезпечуючи найбільшу з можливих якість сервісу відносно кожного з абонентів.

Однак, навіть при переході операторів до роботи своїх мереж на основі концепції IMS не зменшуються їхні витрати на обслуговування, ремонт, заміну та купівлю нового обладнання, адже кількість трафіку, який обертається у світі,

зростає з кожним днем. Універсальним рішенням та виходом з даної ситуації стала хмарна технологія віртуалізації мережевих функцій NFV (Network Functions Virtualization).

В ході дослідження було з’ясовано, що технологія NFV дозволяє оператору позбавлятися від спеціалізованого, пропрієтарного і дорогого в обслуговуванні обладнання, замінюючи його віртуальними пристроями, тобто повністю програмними рішеннями на універсальних серверах. Ті чи інші мережеві функції реалізуються віртуально (програмуються) на універсальному, масовому, отже, недорогому обладнанні, наприклад на стандартних серверах, завдяки чому оператори можуть значно знизити різноманітність використовуваного обладнання, а отже, оптимізувати витрати на його придбання і експлуатацію. Важливо зазначити, що віртуалізація будь-якого з компонентів мережі ніяким чином не впливає на якість надання послуг для кожного з користувачів IMS мережі.

Таким чином було з’ясовано, що для створення віртуалізованої експериментальної моделі та досягнення поставленої мети роботи оптимальним рішенням розгортання IMS у хмарному середовищі є модель платформи як послуги (PaaS). На основі даної моделі обрано і спосіб реалізації IMS у «хмарі» – розділена IMS, або ж split-IMS. Даний спосіб створює можливість до розгортання функціонального елементу мережі розбиваючи його на декілька компонентів і засновується він на принципі багатоетапної подійно-орієнтованої архітектури SEDA.

Проаналізувавши інформація щодо організації та особливостей роботи даної концепції було з’ясовано, що на основі архітектури SEDA створюється програмне забезпечення, реалізація якого в високонавантаженій віртуалізованій інфокомунікаційній мережі дозволяє забезпечити оптимальну роботу кожного з елементів системи. Так, за допомогою шаблонів побудови моделі обробки заявок в даній концепції можна створити специфіковане програмне забезпечення, яке необхідне для вирішення завдань, що постають перед оператором інфокомунікаційних послуг в залежності від його потреб та вимог.

Для реалізації завдань даної роботи та проведення експериментального дослідження і, відповідно, створення програмного забезпечення, для використання було обрано шаблон копіювання, який є досить зручним для перевірки продуктивності змодельованого елементу мережі та ресурсозатратності розгорнутої моделі.

На основі створеного програмного забезпечення було проведено експериментальне дослідження роботи одного із компонентів мережі IMS, який реалізовано у хмарному середовищі, а саме HSS на який надходить велика кількість вхідних заявок про надання послуг для користувачів.

Експериментальну модель було розгорнуто на основі серверу RabbitMQ, який має можливість працювати з багатьма операційними системами і хмарними середовищами, а також не є ресурсозатратним, що являється великою перевагою проведення дослідження при ресурсах, які обмежені домашнім персональним комп’ютером. Дана платформа має свою систему моніторингу, яка повністю адаптована під архітектуру SEDA. Так проводячи експеримент можна відслідковувати кількість згенерованих заявок, заявок що надійшли в чергу до кожного з блоків програми та тих, які обробляються обробником.

При проведенні експериментального дослідження було отримано результати, які затверджують, що обробка вхідного потоку заявок відбувається в повному обсязі. Також у зв’язку з тим, що дослідження проведено у різних режимах було доведено, що при великій кількості вхідних заявок та при відсутності великої черги змодельований компонент мережі встигає обробити всі заявки, що до нього надходять, без простою його роботи.

Отже, виходячи з усього вище сказаного, поставлені завдання, які були поставлені перед початком роботи над магістерською дисертацією, можна вважати виконаними.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Бакланов, И. Г. NGN: принципы построения и организации / под ред. Ю. Н. Чернышова. – М.: Эко-Трендз, 2008. – 400 с.: ил.

[2] Гольдштейн Б.С. Инфокоммуникационные сети и системы. – СПб.: БХВ-Петербург, 2019. – 207 с.: ил.

[3] Б. С. Гольдштейн, А. Е. Кучерявый. Сети связи пост-NGN – СПб.: БХВПетербург, 2014. —160 с.: ил.

[4] Гольдштейн Б.С., Гольдштейн А.Б. SOFTSWITCH – СПб.: БХВ – Санкт-Петербург, 2006.— 368 с.

[5] Demystifying LTE Backhaul //[Електронний ресурс] – режим доступу: https://bitly.su/6Aiqx

[6] Гельгор А.Л., Попов Е.А. Технология LTE мобильной передачи данных: учеб. пособие — СПб.: Изд-во Политехн. ун-та, 2011. — 204 с.

[7] LTE: взгляд изнутри //[Електронний ресурс] – режим доступу: https://habr.com/ru/post/136317/

[8] СТАНДАРТ ПЕРЕДАЧІ ДАНИХ LTE //[Електронний ресурс] – режим доступу: https://dmr.kiev.ua/ua/catalog/lte/

[9] Технология SRVCC в мобильных сетях 4-го поколения //[Електронний ресурс] – режим доступу: https://habr.com/ru/post/200868/

[10] Услуги связи. Мультимедийные телефонные услуги – MMTel //[Електронний ресурс] – режим доступу: itechinfo.ru/content/услуги-связи

[11] MMTel – a standard for multimedia services over IMS. Ericsson, October 2008.

[12] CONVERGED MOBILE AND FIXED SERVICES WITH VIRTUAL IMS. Virtualizing and Monetizing IMS Services for Service Providers. VMWare SOLUTION BRIEF.

[13] Хмарні обчислення //[Електронний ресурс] – режим доступу: http://integritysys.com.ua/solutions/pricatecloud-solution/

[14] Довгаль В.А. Облачные вычисления и анализ вопросов информационной безопасности. Научный журнал «Вестник АГУ» Выпуск 2 (161) 2015.

[15] Peter Mell, Timothy Grance. The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology. September 2011. 7 p.

[16] Хмарні технології. Переваги і недоліки. //[Електронний ресурс] – режим доступу: https://valtek.com.ua/ua/system-integration/it-infrastructure/clouds/cloud-technologies

[17] Giuseppe Carella, Marius Corici, Paolo Crosta, Paolo Comi, Thomas Michael Bohnert, Andreea Ancuta Corici, Dragos Vingarzan, Thomas Magedanz. Cloudified IP Multimedia Subsystem (IMS) for Network Function Virtualization (NFV)-based architectures. International Symposium on Computers and Communications, June 2014. 7 p.

[18] Matthew David Welsh. An Architecture for Highly Concurrent, Well-Conditioned Internet Services. University of California, Berkeley, 2002. 190 p.

[19] Офіційний сайт RabbitMQ //[Електронний ресурс] – режим доступу: https://www.rabbitmq.com/

[20] Принципы работы AMQP & RabbitMQ //[Електронний ресурс] – режим доступу: http://helpers.com.ua/amqp/princypy-raboty-amqp-rabbitmq/

ДОДАТОК А

Код роботи графічного інтерфейсу виконуваної програми

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using RabbitMQ.Client;
using RabbitMQ.Client.MessagePatterns;
using RabbitMQ.Client.Events;
using System.Threading;

namespace ims_seda
{
   public partial class Menu : Form
   {
      IConnection connectionSender;
      IConnection connectionListener;
      Thread[] listenersThreads = new Thread[3];
      Thread[] sendMassagesThread = new Thread[3];
      List<Thread> listenersTypeOneThreads = new List<Thread>();
      List<Thread> listenersTypeTwoThreads = new List<Thread>();
      List<Thread> listenersTypeThreeThreads = new List<Thread>();

      public Menu()
      {
         InitializeComponent();
      }

      private void Menu_Load(object sender, EventArgs e)
      {
         connectionSender = RabbitMQController.GetRabbitConnection();
         connectionListener = RabbitMQController.GetRabbitConnection();
      }

      private void BtnConnect_Click(object sender, EventArgs e)
      {
         try
         {
            if (txtUserName.Text == string.Empty || txtPassword.Text == string.Empty ||
txtHostName.Text == string.Empty)
            {
               connectionSender = RabbitMQController.GetRabbitConnection();
               connectionListener = RabbitMQController.GetRabbitConnection();
```

```
            MessageBox.Show("Підключення завершено");
        }
        else
        {
            connectionSender = RabbitMQController.GetRabbitConnection(txtUserName.Text,
txtPassword.Text, txtHostName.Text);
            connectionListener = RabbitMQController.GetRabbitConnection();
            MessageBox.Show("Підключення завершено");
        }
    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
}

private void BtnSendMessage_Click(object sender, EventArgs e)
{
    try
    {
        for (int i = 0; i < sendMassagesThread.Length; i++)
        {
            sendMassagesThread[i] = new Thread(new ParameterizedThreadStart(SendMassage));
            sendMassagesThread[i].Start(i);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
}

private void BtnStopSendMessage_Click(object sender, EventArgs e)
{
    for (int i = 0; i < sendMassagesThread.Length; i++)
    {
        if (sendMassagesThread[i] != null)
        {
            if (sendMassagesThread[i].IsAlive)
            {
                sendMassagesThread[i].Abort();
            }
        }
    }
}

private void BtnModeTwoStart_Click(object sender, EventArgs e)
{
    try
    {
        for(int i = 0; i < listenersThreads.Length; i++)
        {
            listenersThreads[i] = new Thread(new ParameterizedThreadStart(Listener));
```

```
            listenersThreads[i].Start(i);
        }
      }
      catch(Exception ex)
      {
        MessageBox.Show(ex.ToString());
      }
    }

    private void BtnModeTwoStop_Click(object sender, EventArgs e)
    {
      for (int i = 0; i < listenersThreads.Length; i++)
      {
        if (listenersThreads[i] != null)
        {
          if (listenersThreads[i].IsAlive)
          {
            listenersThreads[i].Abort();
          }
        }
      }

      connectionListener.Close();
      connectionListener = RabbitMQController.GetRabbitConnection();
    }

    private void Listener(object num)
    {
      int type = (int)num;

      switch (type)
      {
        case 0:
          RabbitMQController.RabbitListener(connectionListener, "diplom", "video_call",
"video_call", int.Parse(txtFirstListenerSpeed.Text));
          break;
        case 1:
          RabbitMQController.RabbitListener(connectionListener, "diplom", "internet_service",
"internet_service", int.Parse(txtSecondListenerSpeed.Text));
          break;
        case 2:
          RabbitMQController.RabbitListener(connectionListener, "diplom", "voice_call",
"voice_call", int.Parse(txtThirdListenerSpeed.Text));
          break;
        default:
          RabbitMQController.RabbitListener(connectionListener, "diplom", "video_call",
"video_call", int.Parse(txtFirstListenerSpeed.Text));
          break;
      }
    }

    private void SendMassage(object num)
    {
```

```
        int type = (int)num;

        int count = int.Parse(txtMassagesCount.Text);

        int frequency = 0;

        switch (type)
        {
            case 0:
                frequency = 1000 / int.Parse(txtFirstTypeMassagesFrequency.Text);
                break;
            case 1:
                frequency = 1000 / int.Parse(txtSecondTypeMassagesFrequency.Text);
                break;
            case 2:
                frequency = 1000 / int.Parse(txtThirdTypeMassagesFrequency.Text);
                break;
            default:
                frequency = 1000 / int.Parse(txtFirstTypeMassagesFrequency.Text);
                break;
        }


        if(count == 0)
        {
            while(true)
            {
                switch (type)
                {
                    case 0:
                        RabbitMQController.SendMessage(connectionListener, "diplom", "video_call",
"video_call");
                        break;
                    case 1:
                        RabbitMQController.SendMessage(connectionListener, "diplom",
"internet_service", "internet_service");
                        break;
                    case 2:
                        RabbitMQController.SendMessage(connectionListener, "diplom", "voice_call",
"voice_call");
                        break;
                    default:
                        RabbitMQController.SendMessage(connectionListener, "diplom", "video_call",
"video_call");
                        break;
                }
                count--;
                Thread.Sleep(frequency);
            }
        }
        else if (count > 0)
        {
            do
```

```
            {
                switch (type)
                {
                    case 0:
                        RabbitMQController.SendMessage(connectionListener, "diplom", "video_call",
"video_call");
                        break;
                    case 1:
                        RabbitMQController.SendMessage(connectionListener, "diplom",
"internet_service", "internet_service");
                        break;
                    case 2:
                        RabbitMQController.SendMessage(connectionListener, "diplom", "voice_call",
"voice_call");
                        break;
                    default:
                        RabbitMQController.SendMessage(connectionListener, "diplom", "video_call",
"video_call");
                        break;
                }
                count--;
                Thread.Sleep(frequency);
            } while (count > 0);
        }
        else
        {
            MessageBox.Show("Количество сообщений должно быть не больше 65034 и не
меньше 1");
        }
    }
    private void BtnListenerTypeOneAdd_Click(object sender, EventArgs e)
    {
        listenersTypeOneThreads.Add(new Thread(new ParameterizedThreadStart(Listener)));
        listenersTypeOneThreads[listenersTypeOneThreads.Count - 1].Start(0);
    }

    private void BtnListenerTypeTwoAdd_Click(object sender, EventArgs e)
    {
        listenersTypeTwoThreads.Add(new Thread(new ParameterizedThreadStart(Listener)));
        listenersTypeTwoThreads[listenersTypeTwoThreads.Count - 1].Start(1);
    }

    private void BtnListenerTypeThreeAdd_Click(object sender, EventArgs e)
    {
        listenersTypeThreeThreads.Add(new Thread(new ParameterizedThreadStart(Listener)));
        listenersTypeThreeThreads[listenersTypeThreeThreads.Count - 1].Start(2);
    }

    private void BtnModeThreeStop_Click(object sender, EventArgs e)
    {
        for (int i = 0; i < listenersTypeOneThreads.Count; i++)
        {
            listenersTypeOneThreads[i].Abort();
```

```
        }

        for (int i = 0; i < listenersTypeTwoThreads.Count; i++)
        {
            listenersTypeTwoThreads[i].Abort();
        }

        for (int i = 0; i < listenersTypeThreeThreads.Count; i++)
        {
            listenersTypeThreeThreads[i].Abort();
        }

        listenersTypeOneThreads.Clear();
        listenersTypeTwoThreads.Clear();
        listenersTypeThreeThreads.Clear();

        connectionListener.Close();
        connectionListener = RabbitMQController.GetRabbitConnection();

        //MessageBox.Show(listenersTypeOneThreads.Count.ToString()                    +
listenersTypeTwoThreads.Count.ToString() + listenersTypeThreeThreads.Count.ToString());
    }

    private void Menu_FormClosing(object sender, FormClosingEventArgs e)
    {
        for (int i = 0; i < listenersTypeOneThreads.Count; i++)
        {
            listenersTypeOneThreads[i].Abort();
        }

        for (int i = 0; i < listenersTypeTwoThreads.Count; i++)
        {
            listenersTypeTwoThreads[i].Abort();
        }

        for (int i = 0; i < listenersTypeThreeThreads.Count; i++)
        {
            listenersTypeThreeThreads[i].Abort();
        }

        for (int i = 0; i < listenersThreads.Length; i++)
        {
            if (listenersThreads[i] != null)
            {
                if (listenersThreads[i].IsAlive)
                {
                    listenersThreads[i].Abort();
                }
            }
        }

        for (int i = 0; i < sendMassagesThread.Length; i++)
        {
```

```
            if (sendMassagesThread[i] != null)
            {
                if (sendMassagesThread[i].IsAlive)
                {
                    sendMassagesThread[i].Abort();
                }
            }
        }

        connectionSender.Close();
        connectionListener.Close();
    }
  }
}
```

ДОДАТОК Б

Код виконуваної програми

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using RabbitMQ.Client;
using RabbitMQ.Client.MessagePatterns;
using RabbitMQ.Client.Events;
using System.Windows.Forms;
using System.Threading;

namespace ims_seda
{
   class RabbitMQController
   {
      public static IConnection GetRabbitConnection()
      {
         ConnectionFactory factory = new ConnectionFactory
         {
            UserName = "guest",
            Password = "guest",
            VirtualHost = "/",
            HostName = "localhost"
         };

         IConnection conn = factory.CreateConnection();
         return conn;
      }

      public static IConnection GetRabbitConnection(string userName, string password, string
hostName)
      {
         ConnectionFactory factory = new ConnectionFactory
         {
            UserName = userName,
            Password = password,
            VirtualHost = "/",
            HostName = hostName
         };

         IConnection conn = factory.CreateConnection();
         return conn;
      }

      static private IModel GetRabbitChannel(IConnection connect, string exchangeName, string
queueName, string routingKey)
```

```
        {
            IModel model = connect.CreateModel();
            model.ExchangeDeclare(exchangeName, ExchangeType.Direct);
            model.QueueDeclare(queueName, false, false, false, null);
            model.QueueBind(queueName, exchangeName, routingKey, null);
            return model;
        }

        public static void SendMessage(IConnection connect, string exchangeName, string
queueName, string routingKey)
        {
            IModel model = GetRabbitChannel(connect, exchangeName, queueName, routingKey);
            byte[] messageBodyBytes = Encoding.UTF8.GetBytes("Hello, world!");
            model.BasicPublish(exchangeName, routingKey, null, messageBodyBytes);
            model.Close();
        }

        public static void RabbitListener(IConnection connect, string exchangeName, string
queueName, string routingKey,int speed)
        {
            try
            {
                IModel model = GetRabbitChannel(connect, exchangeName, queueName, routingKey);
                if (speed != 0)
                {
                    model.BasicQos(0, (ushort)speed, false);
                }
                var subscription = new Subscription(model, queueName, false);
                while (true)
                {
                    BasicDeliverEventArgs basicDeliveryEventArgs = subscription.Next();
                    subscription.Ack(basicDeliveryEventArgs);

                    if (speed != 0)
                    {
                        Thread.Sleep(1000 / speed);
                    }
                }
            }
            catch
            {
                //MessageBox.Show(ex.ToString());
            }
        }
    }
}
```